Blockchain Database for a Cyber Security Learning System

Sophia Armstrong
Department of Computer Science, College of Engineering and Technology
East Carolina University

Te-Shun Chou
Department of Technology Systems, College of Engineering and Technology
East Carolina University

John Jones
College of Engineering and Technology
East Carolina University

Abstract

Our cyber security learning system involves an interactive environment for students to practice executing different attack and defense techniques relating to cyber security concepts. We intend to use a blockchain database to secure data from this learning system. The data being secured are students' scores accumulated by successful attacks or defends from the other students' implementations. As more professionals are departing from traditional relational databases, the enthusiasm around distributed ledger databases is growing, specifically blockchain. With many available platforms applying blockchain structures, it is important to understand how this emerging technology is being used, with the goal of utilizing this technology for our learning system. In order to successfully secure the data and ensure it is tamper resistant, an investigation of blockchain technology use cases must be conducted. In addition, this paper defined the primary characteristics of the emerging distributed ledgers or blockchain technology, to ensure we effectively harness this technology to secure our data. Moreover, we explored using a blockchain database for our data.

1. Introduction

New buzz words are constantly surfacing in the ever evolving field of computer science, so it is critical to distinguish the difference between temporary fads and new evolutionary technology. Blockchain is one of the newest and most developmental technologies currently drawing interest. It first attracted attention after the tremendous success of Bitcoin's blockchain protocol. Simply stated, a blockchain is a sequenced structure of data that is secured using some time of cryptographic signature. A blockchain can also be characterized as a decentralized distributed ledger. Each ledger, or block, contains some kind of data, typically data transactions, such as Bitcoin transfers [1]. These ledgers are distributed among a secure blockchain network. In this blockchain network, the transactions must be verified. To ensure high quality security, the network transactions can only be verified if the majority of users in the blockchain approve the

verification. When data is supplied to the blockchain, it becomes almost impossible to tamper with because the transactions are monitored by the members of the blockchain network and everyone has a copy of the recorded transactions. Furthermore, a blockchain can "be defined as a consensus oriented secured distributed public/private ledger which stored data over a peer to peer network" [2], reiterating the point that a blockchain is a type of distributed ledger.

A distributed ledger can be intuitively thought of as a data structure, much like the commonly used linked list, where the "blocks" are in a chain, and each block refers to the block prior to them, functioning much like a reversed linked list. The blockchain is a type of distributed ledger consisting of an unalterable record of transactions that is distributed to all users, eliminating the chance of single point failure. In order to remove the need for a centralized authority to process or validate transactions, the distributed ledger is decentralized, so each record can be viewed and verified by all users. This allows users to audit any transactions that may be inconsistent with what is expected. Given that transaction states are monitored carefully, this system is most suitable to replace databases that involve items that change states frequently [1]. Distributed ledgers can also be thought of as another type of database with the defining characteristic of distributing across multiple sites, computers or even countries, where data is stored in continuous ledgers. Moreover, a blockchain is a database consisting of blocks of data that are 'chained' to the next block using a type of cryptographic signature. This gives the blockchain the ability to be manipulated like a ledger, which can be shared and validated by anyone with permission. But blockchain technology is more than just a distributed database; "it can also set rules about a transaction (business logic) that are tied to the transaction itself. This contrasts with conventional databases, in which rules are often set at the entire database level, or in the application, but not in the transaction [3]." Overall, a blockchain is a type of distributed ledger storage mechanism with cryptographic security features and consensus based algorithms to verify the actions done to the distributed ledgers.

Notably, the distributed nature of blockchain technology suitably provides protection for our student's scores by allowing multiple records of their scores to be securely stored on the machines comprising the blockchain network. A blockchain database will also provide some verification mechanisms, so if a student gets past the firewall and penetrates one of the machines with the database, they will not be able to alter it without a key value. Also, for most blockchain networks, once data is put on the network that entry is immutable, so students will not be able to change already stored data. Additionally, if a student does so happen to break into one of the databases, there will still be copies of the unaltered database distributed among other machines. Our implementation for a blockchain databases used BigchainDB, further details and a software analysis for using this software is discussed in following sections.

This paper is organized as follows: Section 2 discussed how blockchain technology can be used. Section 3 defined the cyber security learning system's data storage needs. Section 4 described our efforts to implement an appropriate decentralized blockchain protocol for the data using BigchainDB. Section 5 concluded the findings of this research and suggests further research to be conducted.

2. Blockchain Technology Applications

The first notable usage for blockchain technology arose from cryptocurrencies, the most well-known being Bitcoin, where the distributed ledger structure allows the currency transactions to be monitored and distributed among many computers on a network without a central authority. Here the transaction data is encrypted using cryptographic algorithms and verified by other computers on the network. Cryptocurrency may have introduced blockchain technology to the industry, but there are many other applications that can benefit from its features including financial, commercial, and even improving transactions in developing countries. In the financial sector, many are taking advantage of the security, decentralization and immutability of this technology to develop private ledgers that cut costs and allow real time market activity to be monitored [4]. On the commercial side of the industry, Factom and Everledger are companies making progress. Factom focuses on securing data, while Everledger is focusing on identifying and verifying items, such as tracking diamonds from mine to market. Another company providing protection for the financial field is Ripple. Ripple is using an unsupervised approach to detecting fraud that is based on a distributed ledger network. Ripple provides a payment network, currency exchange market, and a real time settlement system that allows banks to send international payments across multiple networks faster than other methods. The information sent through the Ripple network is represented by a distributed ledger structure consisting of a collection of transactions, user information, a date and a sequence number that the network stores as a sequence of ledgers. As with most blockchain protocols, new information must be verified, and a consensus must be made between the nodes on the network to identify which transactions modify which ledger [5]. In developing countries, the need for trusted transactions play a large roll for issues such as land registration to secure land ownership titles. Blockchain technology is being used to monitor and ensure only trusted transactions are being made to keep land records honest for underrepresented people in developing countries [4].

Blockchain technology is also being used to protect data, as described in "Decentralizing Privacy: Using Blockchain to Protect Personal Data," where researchers used a blockchain protocol to create an automated data access-control manager [6]. Other potential use cases for blockchain include protecting digital infrastructure from cyberattacks [3] and controlling access to volatile data in web applications [7]. The exponential growth of digital data is posing many threats to data security, consider the rapid formation of the internet of things, but blockchain technology could be the answer the industry is looking for. Mobius, an open source internet of things server platform, is already integrating blockchain into its system to store and secure the real time sensor data. It promises better security than the previous MySQL server with the recently discovered vulnerability to "a deodorization method using SQL injection and remote access, utilizing the transmission method using the http protocol ruling [8]." The configuration suggests to store the sensor data in a blockchain of Ethereum's, an open source virtual currency network to prevent forgery by utilizing Ethereum's security hashing processes of creating public and private keys to secure data transmissions. Ethereum also plans to utilize Smart Contract, which is a method of encryption and authentication used when a user requests data that uses the private keys [8]. Ultimately, the security features and decentralized distributed nature provided by blockchain technology is beneficial to many industries that want to insure secure data and any data transactions that may be necessary.

3. Cyber Security Learning System's Data Architecture

Considering the added security, a decentralized blockchain system provides, it is a good fit to secure the data from our cyber security learning system. In this system, each student participating has ten virtual machines for their defender security learning environments and an additional one for their attacker environment. The system is currently intended for ten students but can be expanded in the future if needed. Every student reads an introduction on the cyber-attacks they are implementing and defending from the other student's implementations. Then they take a quiz on the material before launching a virtual machine for a hands-on experience dealing the with cyber security topics they are now familiar with. As previously mentioned, each student has a total of eleven virtual machines that communicate through a network.

The objective of this learning system is to provide a game-based learning environment, so a score is kept for the students monitoring who successfully break into or protect their virtual environments. As these students are being trained to break into systems, a secure network must be used to check the scores. That network, named BREserver, is connected to all the virtual environments with the sole purpose of checking the environments for attacks and sending updates on the virtual machines. A scoreboard virtual machine then communicates with the BREserver virtual machine through a socket, receiving the updates as they come through. The blockchain database is incorporated into the scoreboard virtual machines, where the database is decentralized on multiple nodes running on different machines/IPs. The blockchain database was implemented using BigchainDB, an open source distributed database technology that incorporates blockchain characteristics.

4. Methodology

4.1. Terminology

Our implementation of a blockchain database used BigchainDB, so our first step in building our database was to understand BigchainDB terminology. The most fundamental structure created by BigchainDB is the BigchainDB Node. A BigchainDB Node is a logic machine running the BigchainDB Server software and other necessary software [9]. Each node must consist of at least the BigchainDB Server, a MongoDB server, Tendermint, and storage for MongoDB and Tendermint [10]. Tendermint is the blockchain core protocol that gives BigchainDB its blockchain characteristics. Additionally, MongoDB is an open source database server/client protocol that operates using a NoSQL querying language [12]. The MongoDB server is used as the backend storage for the data from BigchainDB, providing BigchainDB its database characteristics. Security for the MongoDB is insured "by avoiding DoS attacks at the NGINX proxy layer and by ensuring that MongoDB has TLS enabled for all its connections [13]." A collection of connected BigchainDB nodes forms a BigchainDB cluster, where each node is set up running the same software [10]. BigchainDB nodes can communicate to each other using the networking and consensus protocols implemented by Tendermint. A BigchainDB Network is a cluster of 4 or more BigchainDB nodes [10]. The group of people or organizations who operate the nodes of the BigchainDB Network are referred to as the "consortium, [11]" and should all be equal in the amount of power they have. "Each node has its own local MongoDB database," and each member of the consortium is in charge of diversifying their node. Diversity is important

because “If all the nodes are running the same code, i.e. the same implementation of BigchainDB, then a bug in that code could be used to compromise all of the nodes. Ideally, there would be several different, well-maintained implementations of BigchainDB Server (e.g. one in Python, one in Go, etc.), so that a consortium could also have a diversity of server implementations. Similar remarks can be made about the operating system [9].” The BigchainDB Server’s main functionality is to set up the environment a node will operate with and initiate the local MongoDB and Tendermint instances. This is critical for the creation of a BigchainDB cluster, correctly setting up the first node to ensures all other dependencies should operate successful.

Another fundamental concept necessary for understand how BigchainDB operates is how they represent the data being stored. Data is structured as “assets,” which can be any digital or physical objects [14]. In our case, the assets being stored in BigchainDB are student’s scores. Assets can be generated on the BigchainDB network using the “CREATE” transaction, or the “TRANSFER” transaction. The CREATE transaction creates a new asset and associates it with the desired user who owns this asset. The CREATE transaction intakes the desired characteristics of this new asset and the owner’s credentials. For example, if a user wants to create an asset for the book they are trying to sell, a CREATE transaction will be used and the input would include information such as title, author, publishing date, etc. Along with information about the asset, information about the user owning the asset is inputted when using the CREATE transaction. The user who owns this asset is represented by a public key that is inputted and validates that this specific user owns this asset. The asset is then signed with the user’s private key to further validate initial ownership of the asset. If the public and private key are not associated to the same user, the transaction will not be verified and will not be created. A transaction ID is generated for the asset, containing a hash of all the information created by this initial CREATE transaction. This transaction ID is used in later transactions to reference the asset and the input data associated with the asset. The output for the CREATE transaction states the official owner of the asset, the user who created and signed the asset with their public and private keys and defines that this asset’s ownership cannot be transferred without a signature with the private key of the owner. Ownership of the asset can be transferred to another user with the TRANSFER transaction, that intakes a reference to the output of the CREATE transaction and must contain a signature from the original owner’s private key to validate the user is aware and approving the transfer [14]. A BigchainDB driver is necessary to perform the described CREATE and TRANSFER transactions.

In order to connect and interact with the BigchainDB Network that set up using the BigchainDB Server, a BigchainDB driver is necessary. There are many official drivers and tools available to connect to the BigchainDB Network, but two of the most popular are the Python and JavaScript drivers. The driver requires a BigchainDB Root URL to access this Network and communicate with the node or cluster. There are three possible cases for determining what the BigchainDB Root URL is [15]. Case one is if a BigchainDB node is running locally, and if the BIGCHAINDB_SERVER_BIND setting wasn’t changes, then the BigchainDB Root URL is ‘https://example.com:9988.’ Case two involves a cluster hosted by someone else, so they will provide their set BigchainDB Root URL. Case three is if the node is running the Docker-based dev setup, and the connections is from the BigchainDB driver that comes with this setup, then the BigchainDB Root URL is 'http://bdb-server:9984' [16]. There are other variations for

choosing the correct BigchainDB Root URL, but these are the most typical. The BigchainDB Root URL can also be manually set when creating a node but editing the BIGCHAINDB_SERVER_BIND setting [16]. Overall, BigchainDB is a blockchain database consisting of a BigchainDB Network, initialized by the BigchainDB Server, and accessed/manipulated using one of the BigchainDB drivers.

4.2. Installation

For our BigchainDB implementation, we used an Ubuntu 18.04 virtual machine to create the first node in our cluster. We initially began installation using an Ubuntu 16.04 virtual machine, but Python3.6 is required for BigchainDB, and this version is not automatically installed. There are installation methods for Python3.6 on Ubuntu 16.04, but when calling python3 the automatic response is to use the Python3.5 version that is installed. We successfully set python3 to use 3.6 instead of 3.5 by using the "sudo update-alternatives" command, then made sure our system was up to date using the "sudo apt update" and "sudo apt full-upgrade", as recommended by the installation directions for setting up a node [10]. Then we ran "sudo apt install -y python3-pip libssl-dev" to install the required packages. We checked that pip3 was using Python3.6, and it was, so we continued and ran "sudo pip3 install bigchaindb==2.0.0a6" to install the latest version of BigchainDB. However, we ran into an error during this execution, the Python.h file could not be found, which corrupted the build of the remaining dependencies. We suspected this error was due to the installation of Python3.6 and after many hours how trying to resolve the issue, we were unsuccessful. We decided to instead use an Ubuntu 18.05 virtual machine, as the documents provided were tested on an Ubuntu 18.04 LTS Virtual Machine on Microsoft Azure [10].

Using the Ubuntu 18.05 virtual machine, installation was successful. Following the instructions for Member set up, each member set up its own node consisting of the "BigchainDB Server, Tendermint Core, MongoDB, and configuring the firewall. [10]" To set up the firewall, we accepted inbound connections for ports 9984, 9985, 26656, and also added port 22 to continue access to the machine via SSH. To configure the BigchainDB Server, we ran bigchaindb configure and we bind the API Server to 0.0.0.0:9984 to expose the API to the public. The other configure questions we kept the default, but may need to edit these later. Every node in the network is identified by a generated public key, private key, hostname, and node id in the form of the triplet: <hostname, node_id, public_key>. Every member of the network must create and securely store their node's private key, then share the <hostname, node_id, public_key> triplet with all members. We did this using the "tendermint init" command, that stored the nodes public_key in the file .tendermint/config/priv_validator.json [10]. An example of this json file is shown in Figure 1.


```
{
 "address": "5943A9EF6285791A504918E1D117BC7F6A615C98",
 "pub_key": {
      "type": "AC26791624DE60",
      "value": "W3tqeMCU3d4SHDKqrwQWTahTW/wpieIAKZQxUxLv6rI="
 },
 "last_height": 0,
 "last_round": 0,
 "last_step": 0,
```

```
  "priv_key": {
       "type": "954568A3288910",
      "value":
"3sv9aExgME6MMjx0JoKVy7KtED8PBiPcyAgsYmVizslbe2p4wJTd3hIcMqqvBBZNqFNb/CmJ4gAplDFTEu/qsg=="
  }
}
```


Figure 1. An example of json file

To get the node's node_id, we ran "tendermint show_node_id" and for the hostname we use localhost. We shared the node_id, pub_key.value and hostname of the nodes with all other members. In order it insure high quality security, "each Member should take extra steps to verify the public keys they receive from the other Members have not been tampered with, e.g. a key signing party would be one way [10]." The Coordinator, the person in charge of connecting the nodes together, received the necessary data that was previously discussed, and combined the data in the .tendermint/config/genesis.json file. The genesis.json file now contained all the data for the nodes in the network, and the Coordinator shared this file with all the Members. We have three other nodes in our network, for a total of four nodes, and they are all running on Ubuntu 18.04 virtual machines. Our Members in charge of the other nodes then verified the genesis.json file contained the correct public keys and stored a local copy of the genesis.json file in the .tendermint/config directory. All Member now have the same chain_id, gensis_time, and validators. Additionally, each Member edited the .tendermint/config/config.toml to set the creat_empty_blocks to false and add the Member information to the persistent_peers variable. The final step in setting up the BigchainDB network was for each Member to run "bigchaindb start" and "tendermint node" to run the network on the foreground. To keep the nodes running after logout, "nohup", "tmux" or "screen" can be used [10]. To start the network immediately following the operating system's boot, startup scripts for BigchainDB and Tendermint are recommended and we plan to write these. Our BigchainDB installation is completed but applying our BigchainDB database to secure the student's data is still in progress.

4.3. Our Application

After understanding and installing BigchainDB, utilizing the database to secure the student's data was the next task. While using some kind of encryption key to lock our database would suffice to secure the student's data, our goal was to distribute the database among multiple sources as an additional precaution. With this in mind, blockchain technology supplies encryption feature as well as distributed characteristics, providing to be a good fit for our security needs. As previously mentioned, each student has ten defender virtual machines and one attacker virtual machine. The log file monitored each defender virtual machine for attacker's success and outputs a status report on itself and each attacker. TLS and SSL are used to secure the data before coming to our database. We initially used a python script to read in the log file and updated the scoreboard virtual machine database, which was a MongoDB database, according to the IP address of the successful attack. So, to integrate our distributed BigchainDB databases, we plan to use the CREATE transaction to push the data into the blockchain network, storing a copy of the updated student's score on each node in the network. Once an asset is pushed to the database, that record becomes immutable and can only be updated through another transaction, but a copy of that record remains in the chain of events on our blockchain network [17]. We will import the BigchainDB Python-Driver to our existing python script and create an

asset model to describe the students' score. Every time a student's defender virtual machine is compromised, a TRANSFER transaction for that student's score field will deduct point from their score and transfer points to the successful student attacker. Our asset mode looks like Figure 2.

```
student_score = {
        'data': {
           'ip_address': "172.20.1.252"
            'score': 5
             'message': "Password set to password"
        }
}
```

Figure 2. The asset model

Our transactions operates similarly to the example described earlier with Alice and Bob's Game boy transaction model. Applying this blockchain database software insures student's scores are correctly modified as they cannot be updated without public and private key signatures. These transactions are recorded on the blockchain network and distributed among four machines for another layer of protection. Coupled with the distributed nature of BigchainDB, the immutability of the data, and the private key signatures needed to update students' scores, BigchainDB provides suitable security for our students' scores.

5. Conclusions

Emerging technology is constant surfacing, then perishing in the computer science field, but blockchain is here to stay. Blockchain techniques adapt well to the security needs for the rapidly growing data production, due to the cryptographic signatures and consensus algorithms that provide proof of ownership and validity distributed across many users. BigchainDB is one of the many companies making use of this distributed ledger methodology and provided suitable protection for our students' score data. Further research must be conducted to finish our application, but the infrastructure is in place and our future strategy has been determined. In conclusion, blockchain technology is more than just a framework for cryptocurrencies, it is applicable to many diverse initiatives and should be further investigated to improve existing technical procedures.

Acknowledgements

This research is based upon work supported by the Secure &Trustworthy Cyberspace (SaTC) Program of the National Science Foundation under Grant Number 1723650. The authors are grateful to the support of Department of Technology Systems in the College of Engineering and Technology at East Carolina University.

References

[1] S, McLean and S. Deane-Johns, "Demystifying Blockchain and Distributed Ledger Technology - Hype or Hero?" *Computer Law Review International,* vol. 17, *(4),* pp. 97-102, 2016. Available: ProQuest, http://search.proquest.com. [Accessed: Jun. 9, 2018].

[2] R. D. Camino, R. State, L. Montero and P. Valtchev, "Finding Suspicious Activities in Financial Transactions and Distributed Ledgers," *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, New Orleans, LA, 2017, pp. 787-796, 2017. [Online]. Available: IEEE Xplore, http://www.ieee.org. [Accessed: Jun. 15, 2018].

[3] G. O. for Science, M. Hancock, and E. Vaizey, "Distributed Ledger Technology: beyond block chain." *The UK Government Chief Scientific Advisor,* Jan 2016. [Online]. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf. [Accessed: Jun. 19, 2018].

[4] S. Underwood, "Blockchain Beyond Bitcoin," *Association for Computing Machinery. Communications of the ACM,* vol. 59, *(11),* pp. 15, 2016. Available: ProQuest, http://search.proquest.com. [Accessed: Jun. 15, 2018].

[5] R. D. Camino, R. State, L. Montero and P. Valtchev, "Finding Suspicious Activities in Financial Transactions and Distributed Ledgers," *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, New Orleans, LA, 2017, pp. 787-796. Available: IEEE Xplore, http://www.ieee.org. [Accessed: Jun. 16, 2018].

[6] G. Zyskind, O. Nathan and A. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," *2015 IEEE Security and Privacy Workshops*, San Jose, CA, 2015, pp. 180-184. Available: IEEE Xplore, http://www.ieee.org. [Accessed: Jun. 16, 2018].

[7] L. Xu, L. Chen, N. Shah, Z. Gao, Y. Lu, and W. Shi. "DL-BAC: Distributed Ledger Based Access Control for Web Applications," *WWW '17 Companion Proceedings of the 26th International Conference on World Wide Web Companion,* Pages 1445-1450, April 3-7, 2017. [Online]. Available: ACM Digital Library, http://dl.acm.org/. [Accessed: Jun. 15, 2018].

[8] J. H. Jeon, K. Kim and J. Kim, "Block chain based data security enhanced IoT server platform," *2018 International Conference on Information Networking (ICOIN)*, Chiang Mai, 2018, pp. 941-944. Available: IEEE Xplore, http://www.ieee.org. [Accessed: Jun. 16, 2018].

[9] "Terminology," 2018. [Online]. Available: https://docs.bigchaindb.com/en/latest/terminology.html. [Accessed: Jun. 18, 2018]

[10] "How to Set Up a BigchainDB Network," 2018. [Online]. Available: http://docs.bigchaindb.com/projects/server/en/latest/simple-deployment-template/network-setup.html. [Accessed: Jun. 18, 2018]

[11] "Bigchaindb 2.0 the blockchain database," May 2018. [Online]. Available: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf. [Accessed: Jun. 18, 2018].

[12] "The mongodb 4.0 manual," 2018. [Online]. Available: https://docs.mongodb.com/manual/. [Accessed: Jun. 18, 2018].

[13] "Architecture of a BigchainDB Node Running in a Kubernetes Cluster," 2018. [Online]. Available: http://docs.bigchaindb.com/projects/server/en/master/k8s-deployment-template/architecture.html. [Accessed: Jun. 18, 2018].

[14] "Key concepts of BigchainDB," 2018. [Online]. Available: https://www.bigchaindb.com/developers/guide/key-concepts-of-bigchaindb/. [Accessed: Jun. 18, 2018].

[15] "Determine the BigchainDB Root URL," 2018. [Online]. Available: https://docs.bigchaindb.com/projects/py-driver/en/latest/connect.html. [Accessed: Jun. 18, 2018].

[16] "BigchainDB Python Driver," 2018. [Online]. Available: https://github.com/bigchaindb/bigchaindb-driver#bigchaindb-python-driver. [Accessed: Jun. 18, 2018].

[17] "How BigchainDB is Immutable," 2018. [Online]. Available: https://docs.bigchaindb.com/en/latest/immutable.html. [Accessed: Jun. 18, 2018].

Biography

SOPHIA ARMSTRONG is a senior undergraduate student in the Department of Computer Science at East Carolina University. She is working as the lead software developer for a National Science Foundation founded grant number 1723650 where she programs the user interface. She is graduated Spring 2019 where she plans to enter the workforce as a software engineer.

TE-SHUN CHOU is an Associate Professor in the Department of Technology Systems at East Carolina University. He received his Bachelor degree in Electronics Engineering at Feng Chia University and both Master's degree and Doctoral degree in Electrical Engineering at Florida International University. He serves as the program coordinator of the Master program in Network Technology for the Department of Technology Systems and the lead faculty of Digital Communication Systems concentration for the Consortium Universities of the Ph.D. in Technology Management.

JOHN JONES is currently an Instructional Technology Consultant with the College of Engineering and Technology at East Carolina University. He received his Bachelor degree in Computer Science from Ohio University and a Master's degree in Technology Management / Information Security from East Carolina University. He has worked in the IT industry over 20 years in varied roles such as software design, IT director, IT security, infrastructure management, systems administration, Internet Services and part-time faculty.