FACULTY OF SCIENCE & TECHNOLOGY

BSc (Hons) Forensic Computing and Security
May 2021

THREAT-CHAIN:
A Peer-to-Peer Standardized Cyber Threat Intelligence Sharing Application

by

Frazer Chard

Faculty of Science & Technology
Department of Computing and Informatics
Final Year Project

ABSTRACT

Current methods of cyber-threat dissemination are unstructured and are only generally shared with peers in which there is already some level of personal trust. This causes serious limitations in regards to the scope of the cyber threat landscape to which organisations are privy. There have been many different efforts to solve these problems individually with several different threat intelligence expressions to solve the issue of interoperability, as well as the use of the blockchain to create an immutable and trusted data-sharing platform with a central data store.

Blockchain technology enables data to be stored in an immutable manner over a distributed ledger to provide non-repudiation of data uploads, with a heavy focus on the integrity and availability of the data. The use of distributed data storage ensures that not only is the data available at all times but through content-based addressing, the data cannot be altered.

This report presents a prototype for semi-decentralized federated Quorum blockchain-based application that enables the upload, conversion and retrieval of cyber threat intelligence to structured threat expressions, stored in a decentralised and distributed data store, enabling only the permissioned users to interact with it. It was found that although the integrity and availability of the process were improved immensely, the data confidentiality was found to be poorer due to a lack of privacy-focused features.

DISSERTATION DECLARATION

I agree that, should the University wish to retain it for reference purposes, a copy of my dissertation may be held by Bournemouth University normally for a period of 3 academic years. I understand that once the retention period has expired my dissertation will be destroyed.

Confidentiality

I confirm that this dissertation does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or sex life has been anonymised unless permission has been granted for its publication from the person to whom it relates.

Copyright

The copyright for this dissertation remains with me.
Requests for Information
I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act

Signed: *FrazerC*
Name: Frazer Chard
Date: 01/03/2021
Programme: Forensic Computing and Security

Original Work Declaration

This dissertation and the project that it is based on are my own work, except where stated, in accordance with University regulations.

Signed**:** *FrazerC*
Name: Frazer Chard
Date: 01/03/2021

ACKNOWLEDGMENTS

I would like to thank Muntadher Sallal for supervising me with my project. I would also like to say thank you for Cayo Fletcher-Smith for all the continued support throughout the last three years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 PROBLEM DEFINITION & CONTEXT

In recent years the occurrence of cyber attacks has been continually growing, as seen with the proportion of companies reporting cyber attacks in 2020, with 5% increase from 38% to 43%. These reported attacks also show that threat actor persistence is growing, with 47% of these organisations having to defend against the same actors more than 5 times and 33% of the organisations having to fend off the malicious actors more than 25 times (Wharton, 2021). Threat actor behaviour is perpetually growing more evasive in an attempt to bypass traditional security solutions. This is evidenced by Carbon Blacks 2020 report, in which after an analysis of malware samples it was found that over 90% of the sample set contained defensive evasion techniques to avoid detection (Carbon Black, 2020).

Within the cybercrime enterprise, the techniques that threat actors use do not follow practices defined within a regulated system, which permits complete freedom in their attack method attributing to less predictable approaches, and a high level of tenacity and resourcefulness(Sahrom Abu et al. 2018) .In contrast, the structured system of corporate enterprise is defined by strict regulatory legislation and frameworks, which hinders the speed at which new security techniques can be incorporated into a system. Most small-to-medium enterprises (SMEs) cannot keep up with adversaries due to a constant and rapid evolution in the tools, tactics and procedures (TTPs) being deployed.

When traditional methods of security are no longer effective against the new methods of attack, identifying the incidents becomes much more difficult. A big part to play in this is a severe lack of understanding of the current cybersecurity landscape. In an attempt to keep pace with the ever-evolving TTPs, Threat Intelligence (TI) is used as a way to enable organisations to identify, assess, monitor and respond to cyber threat incidents (NIST Special Publication, 2016). TI enables organisations to take a proactive approach when it comes to the detection and prevention of further attacks.

In 2016, AlienVault conducted a survey at Black Hat which found that the majority of the respondents relied most on their own internal detection processes for threat intelligence, minimizing the sources of valid threat intelligence to just one system (Alien Vault, 2016). Organizations are encouraged to collaboratively share their own TI as a mutually beneficial method to mitigate future attacks, enabling a wider view of the current cybersecurity landscape but without trust, collaboration can be challenging due to the sensitive nature of the data, as can be evidenced by the AlienVault survey, which showed that generally people were much less willing to use crowdsourced threat intelligence platforms. Even if the organizations were willing to collaborate, there are serious interoperability issues when it comes to the dissemination of threat intelligence as there is no standard format that TI producers can agree on. Organizations, such as MITRE, are working on developing standardised threat intelligence formats (Barnum, 2014).

Distributed technologies have the capability to disrupt the way many traditional corporate enterprises can function and collaborate together. A large component of this is blockchain technology which, through the use of secure cryptographic protocols,high level access control and a multitude of consensus mechanisms, enables the potential for a trusted system with unknown agents.

Through the use of peer-to-peer (P2P) data storage, the rate of data transmission should be faster than typical centralised storage solutions and the immutable manner in which data is recorded to a blockchain ensures freedom from data tampering, along with an added layer of non-repudiation for the data producer (Decentralized storage transfer speeds faster, 2018) (The Immutability of Cryptocurrency: Part 1 - Concerning Immutability, 2016). All of these features of the blockchain are properties which lend themselves well to the issues within modern threat intelligence sharing.

1.2 PROPOSED SOLUTION

The proposed solution is to design and create a prototype for the functional aspect of a cyber threat intelligence platform using distributed ledger technology to improve on the confidentiality and integrity of traditional threat intelligence platforms. The application will allow a small syndicate of authorised nodes to format, upload and consume structured threat intelligence using a data store implemented alongside distributed ledger technology to ensure data immutability and integrity; Members of the threat intelligence syndicate are not required to trust the other members of the network, but just have to trust in the technology underpinning it. The distributed ledger will ensure that only nodes associated with the syndicate will be capable of uploading and consuming the threat intelligence stored in the distributed storage system.

1.3 CORE OBJECTIVES

Objectives	Success Criteria
Conduct a review of the literature surrounding distributed ledgers, storage and threat intelligence	Identify relevant sources to enhance knowledge on: • Methods to collect and disseminate threat intelligence • Secure transactions using distributed technologies • Storage implemented distributed ledger technologies
Create a distributed application that will allow users to interact with a store of cyber threat intelligence	Users of the application will be able to: • Enter cyber threat data into the system • Access threat data that other users have uploaded • Store threat data in a distributed manner
Integrate distributed ledger technology into the app for immutable data production.	Users will be able to: • Access their node on a distributed ledger network • Store a record of transactions to the ledger
Enable users to convert threat data into a structured format	Users should be able to Manually enter threat data • Convert data to a structured threat intelligence expression.
Users will be able to query threat intelligence	Users should be able to Query threat intelligence stored via the ledger to verify the legitimacy of the data.

1.4 PROJECT OVERVIEW

The remainder of this document is organized into the following sections and appendices:

- **Section 2** - Background and Literature Review – Reviewing existing literature surrounding the current threat intelligence landscape, and how cyber threat information is best shared
- **Section 3 -** Software Development Frameworks – An in depth analysis about the development methodologies considered for this project.
- **Section 4** – Specification – A technical analysis analysis of the potential alternatives and selection of the required specification
- **Section 5** - Design – The inital designs and how they formed the final solution
- **Section 6** - Implementation – Describes app features and how it works
- **Section 7** – Testing & Results. Testing requirements, and presenting the results.
- **Section 8** – Conclusion. Concludes the project, including a summary of achievements and an evaluation of the aims and objectives.
- **Section 9** - Further Works. Provides an insight into how this artefact could be progressed.

2 BACKGROUND AND LITERATURE REVIEW

2.1.1 CYBER THREAT INTELLIGENCE DEFINITIONS

With the current evolution in the sophistication of the tools, techniques and procedures there is still no standard definition for certain terms within the current threat landscape. This is evident with terms like cyber attack and cyber threat, or threat information and threat intelligence being used interchangeably throughout the information security community (Scarfone, 2015). It is important for actors in the information security community to comprehend the differences between basic terms that are used extensively and interchangeably.

According to the US Department of Homeland Security's broad definition (Cyber Threat Source Descriptions | CISA, 2005), cyber threats "*refer to persons who attempt unauthorized access to a control system device and/or network using a data communications pathway",* which implies that threats are the act ors performing an attack. This is in contrast to the National Institute of Standards and Technology's (NIST) multitude of definitions (Glossary | CSRC, n.d.), which although vary in content, all follow the core idea that a cyber threat is a circumstance or event with the potential to adversely impact operations, assets and individuals, while an attack is defined as the situation in which a malicious actor is "*targeting an enterprise's use of cyberspace for the purpose of disrupting or maliciously controlling a computing environment, destroying the integrity of data or stealing controlled information." (Glossary | CSRC, n.d.).* Through these, we can see that the general definition for cyber-threat is the condition where malicious activity can happen and a cyber attack is the targetted use of that condition.

There is similar ambiguity between the terms threat intelligence, threat information and threat data. Figure 1 shows the relationship between these three terms, and how they lead to the procurement of actionable threat intelligence, enabling consumers to make informed decisions about how to mitigate cyber threats and prevent cyber attacks (Joint Chiefs of Staff, 2017) .

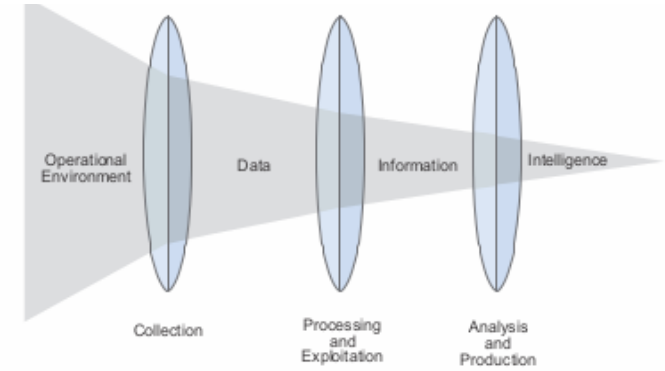


Figure 1: Relationship of Data, Information and Intelligence

The process of procuring actionable threat intelligence starts with a collection of data from an operational environment, which is comprised simply of basic, unrefined and generally unfiltered information. The next stage is the processing, aggregation and organization of the data in a manner that provides context. This leads to the creation of threat information and enables some form of future human analysis. Cyber threat intelligence takes aspects from the analysis of both cyber threats and threat information to provide actionable intelligence, which is evidenced by Brown et al who defines threat intelligence as complex cyber threat information that has been acquired or inferred through the analysis of existing information (Brown, Gommers and Serrano, 2015). This definition is complemented by the idea that Schoeman put forward, in that human analysis of information is necessary as tools and data feeds cannot provide actionable threat intelligence without local contextualization of the system (Schoeman, 2015).

2.1.2 THREAT INTELLIGENCE SOURCES

During the AlienVault survey taken at BlackHat 2016 (Alien Vault, 2016), it was found that 66% of respondents most relied upon threat intelligence from their internal detection processes. The next most reliable source, with 48% agreeing, is from trusted peers, which is commonly shared on an ad-hoc basis with heterogeneous ontology, being communicated via insecure transmission methods, including email exchange and phone calls (Cichonski, 2012). The third most relied upon method, with 44% of the respondents agreeing, is through paid subscription services, such as IBM X-Force Exchange (IBM X-Force Exchange, n.d.) or Mandiant Advantage (FireEye, n.d.) which often provide large volumes of threat data through the use of APIs and a single broadcasting data centre which could make for a very slow system when in high demand and limits the potential for scaling (Alien Vault, 2016). 38% of them relied on government agencies, 37% through crowdsourced platforms and 28% from blogs and forums.

In the same survey, it was found that 56% of the respondents share threat intelligence with trusted peers, and only 15% shared the data with crowdsourced platforms. The key issue with crowdsourced platforms, which have the potential to increase the breadth of cyber threat intelligence sources, is the inherent lack of trust between users when sharing sensitive intelligence and the data quality issue. Managed services will generally ensure that data quality exists within certain parameters by using service level agreements, and when the data originates from a mix of organisations without these agreements, there is the potential for inaccurate, misleading or invalid data being made available (Amoroso, 2011).

2.1.3 THREAT INTELLIGENCE STANDARDS

In recent years, there have been efforts from security researchers to represent threat information in a standardised format to be used across different threat intelligence domains as opposed to the heterogeneous ad-hoc solutions that are commonly used among trusted peers (Cichonski, 2012). This research has led to a vast number of ontologies that intended to solve the issue of interoperability between threat intelligence consumers. Notable examples include NIST's SCAP, Fireye's IOC tool suite and Mitre's range of formats, including CybOX, STIX and TAXII (Kampanakis, 2014). A key advantage to these standards is being able to analyse intelligence from varied sources, all formatted using homogenous ontology to enable better situational awareness of the entire environment.

The viability of these ontologies is wholly dependant on the use case and the required implementation. NIST's SCAP framework is an XML based framework that was developed primarily to provide Federal organizations with a way to show compliance with mandated security requirements (Rayborn, n.d.). OpenIOC is similar in that it is an XML schema, but is simply used to share the description of technical characteristics that identify a threat, attackers methodology or other indicators of compromise (IOC-Finder, n.d.).

MITRE created a range of threat intelligence frameworks, starting with CybOx which was an XML language schema used to encode and communicate high-fidelity information about cyber incidents and was targetted as a flexible solution for all use cases (Mitre, 2012). Following this, STIX was created, which is a structured language and serialization format for the specification, capture, characterization and communication of standardized CTI.

STIX 1.0 utilized an XML schema, that directly leveraged the CybOX schema, while also loosely leveraging CAPEC™, MAEC™ and CVRF schemas (Barnum, 2014). MITRE later developed STIX 2.0 in 2017, which introduced a JSON serialization schema to represent the threat information, with CybOX objects now being referred to as STIX domain objects. The incorporation of JSON into STIX enables a wide range of features that allow for a clear representation of the semantics of cyber threat intelligence information (Cyber Threat Intelligence Technical Committee, n.d.).

2.1.4 THREAT INTELLIGENCE SHORTCOMINGS

There are still many challenges when it comes to sharing threat intelligence. Trust between participants is a key issue. As mentioned before, during the AlienVault survey at BlackHat (Alien Vault, 2016), it was found that 48% of the respondents relied most upon trusted peers for their threat intelligence and that 56% of them shared the discoveries of threats with trusted peers. This shows that there is an inherent trust between the peers sharing the data on a mass scale, with over half willingly sharing threats to peers and just under half relying most upon this same form of intelligence from their peers. The same survey shows that only 40% of the respondents trusted the data that came from their 'trusted' peers, which means that producers are most concerned with the need to trust the identity of the consumer, and not necessarily the quality of the data.

The dissemination of threat intelligence can also bring about potential legal and privacy issues relating to the methods of information distribution of highly sensitive organizational data. Organizations tend to be less willing to share threat information as the information could negatively reflect on their brand (CTI and lessons from law enforcement, 2016). The disclosure of this information in an untrustworthy environment may result in legal violations causing not only financial losses but reputational damage as well. This can lead to organizations becoming hesitant in sharing their intelligence due to the fear of potential repercussions (Amoroso, 2011).

Given the expanding nature of the security landscape, threat intelligence needs to be interoperable and easy to disseminate to other consumers to enable efficient analysis with the context of the surrounding environments. Unstructured formats, such as email, text documents etc. are suited for high-level reports and ad-hoc exchanges to be consumed by security personnel, but not suited for the critical exchanges of indicators. For these indicators, using standardised data formats enables faster transmission and reduces the need for human interaction (NIST Special Publication, 2016).

2.2 DISTRIBUTED LEDGER TECHNOLOGY

2.2.1 BLOCKCHAIN TECHNOLOGY BASICS

In October 2008, Satoshi Nakamoto released the Bitcoin white-paper, which envisioned a purely peer-to-peer version of electronic cash, known as crypto-currency, that can be sent from one party to another without the need for any middlemen to oversee the transaction. Bitcoin was the first conceptualized decentralized blockchain, which implemented a proof-of-work consensus protocol to maintain the state of the ledger (Nakamoto, 2008).

The term blockchain is synonymous with distributed ledger and although Bitcoin is the most well-known blockchain application, there is a wide range of domains that are researching into and beginning to create distributed applications (dApps) that utilize and build upon the core blockchain technologies to improve upon key features in their sector, including but not limited to: supply chain & logistics (Tian, 2016), education and healthcare (Hoy, 2017), Internet of Things (IoT) (Zhang and Wen, 2015) and even in wireless mobile communications (Li, Wang and Wu, 2020).

Blockchain is a composition of different technologies that are used within a decentralized P2P network of nodes to maintain security, transparency and reliability for storing continuously growing data in a distributed public ledger. The data being transacted through the blockchain involves information about asset owners and any transactions involving a change of ownership of said assets, being stored on the distributed ledger using public-private key cryptography to ensure non-repudiation (Warburg, 2016). The way blockchain networks work is that whenever a transaction is entered into the P2P network, the data is sent to every participating node to validate it using an agreed-upon consensus protocol. If the nodes agree on the legitimacy, a new block is created containing information about the transaction and is then appended to the end of the chain.

A key feature of blockchain technology is the immutability that the ledger has, as each new block will contain a cryptographic hash link of the previous block as well as a time-stamp, meaning that a block cannot be tampered with once it has been validated and appended, ensuring the integrity of the data. Following this, the state of the blockchain is updated to every participating node which removes the potential for a central point of failure in situations where a single node is taken down. As the rest of the nodes in the network maintain a copy of the ledger, they will update the downed node with the valid ledger when it comes back online, enabling ad-hoc connectivity to the network (Buterin, 2015).

2.2.2 PUBLIC, PRIVATE AND FEDERATE BLOCKCHAIN

Common characteristics that are seen across both public and private blockchains are the requirements of incentives to participate in the network. Generally, there needs to be some form of

mutual benefit for participating in the network, whether that's financial gain or for full audibility and traceability on past events (Homan, Shiel and Thorpe, 2019).

A public blockchain is permissionless, meaning that anyone can join the network and fully participate with every element of the blockchain. A public blockchain is fully decentralized, meaning that there is no central authority controlling the data, and once the data has been validated and appended onto a public blockchain, it is not possible to alter the blocks but it can be viewed by everyone (Massessi, 2018). Bitcoin and Ethereum are popular examples of public blockchains (Buterin, 2015).

Private blockchains, also known as permissioned blockchains, work on the concept of access control to restrict who can participate in the network and requires permissions to join. Generally, these blockchains are controlled by either a single person or organization and only those participating will know the data on the blockchain. Permissioned blockchains are generally used when private data needs to be stored and should only be accessible to those with the correct authorisation (Massessi, 2018). Hyperledger Fabric is a notable example of a private blockchain (Hyperledger Fabric, n.d.).

Federated blockchains, also known as consortium blockchains, work similarly to private blockchains. The demarcation between the two is that in a federated blockchain there is no sole organization influencing the control of the network, but multiple entities contributing to a decentralized system. This enables a group of organizations to contribute to a network with a mutually beneficial incentive (Hybrid And Federated Blockchain Networks, 2020).

2.2.3 SMART CONTRACTS

The conceptual idea for programmable electronic "smart contracts" came out nearly 25 years ago, with the core principle of combining protocols with user interfaces to formalize and secure relationships over networks (Szabo, 1997). With regards to blockchain networks, a smart contract is a piece of software designed and implemented in high-level contract-oriented languages, that can be automatically executed by nodes on a network when either a certain condition has been met, or through manual operation. Smart contracts have been integrated into many blockchain networks, including Ethereum and serve as the core aspect for distributed applications that run on blockchain technology (Wood, 2021).

Smart contracts provide support for many different aspects of the exchange, such as property, insurance, crypto-currencies, consents or any other asset that carries some form of value (Cook and Lee, n.d.). The implementation of smart contracts enables the exchange of assets without the possibility of censorship, downtime, fraud and no requirement for a central third party to oversee the transaction (Buterin, 2015). Most contracts rely on a traditional web application front-end to provide a user interface for the functions, and these functions are executed by users sending transactions from their public account address to the address that the smart contract has been deployed to (Cook and Lee, n.d.).

2.2.4 ETHEREUM FRAMEWORK

Following the release of Bitcoin, many more blockchain technologies started to emerge, most notably Ethereum, which in 2015 was presented as the next generation of blockchain technology. Ethereum, similar to Bitcoin, has its cryptocurrency associated with it called Ether and another type of currency called gas, which is the fundamental unit of computation for transactions,

introduced to prevent accidental/hostile infinite loops or other computational wastage (Buterin, 2015). Ethereum built upon the core concept of distributed public ledgers for more than just a store of crypto-currencies and offered an environment to create distributed applications that will enable the use of smart contracts for transactions to occur on the blockchain network (Wood, 2021).

Rather than being a distributed ledger, Ethereum is a distributed state machine that holds not only all accounts and the balances associated but also a machine state which can change from block to block according to a set of pre-determined rules. The rules of changing state from block to block are performed by the Ethereum Virtual Machine (EVM), which is a sandboxed virtual stack embedded within each node, responsible for executing smart contract bytecode (Buterin, 2015).

2.2.5 SOLIDITY & THE ETHEREUM VIRUAL MACHINE

Smart contracts are treated like an Ethereum account, meaning they have a balance, public and private key addresses and Ether can be transferred to and from it. When the state of the smart contracts has been updated, a new block will be created to record the change of state. Ethereum smart contracts are typically written in high-level contract-oriented languages, like Solidity, and then compiled into EVM bytecode to be executed by the EVM (Buterin, 2015).

Solidity is an imperatively typed language, similar to JavaScript, used to create smart contracts that will be converted into low-level bytecode instructions through the use of a compiler like Remix. Remix utilizes all aspects of smart contract deployment including an IDE to develop source code with, the ability to deploy contracts to the blockchain, run transactions to the deployed contracts and an integrated compiler to convert the high-level Solidity code into the required low-level EVM bytecode (Remix IDE, n.d.).

Smart contracts themselves can be utilized as the back-end of decentralized applications. The client-side front-end aspect of distributed applications can be implemented through many different front-end languages, including HTML, JavaScript, React and CSS. The back-end of these applications generally implements the web3 framework, which was developed by the Ethereum foundation to assist developers in building client-side distributed applications that can directly interact with the Ethereum blockchain, including both a JavaScript and a Python implementation of the API. Using either of the web3 APIs alongside a JSON-RPC library provides a convenient interface for nodes to interact with the application (web3py, n.d.). Many different development frameworks will help with the whole creation of distributed applications, such as Truffle or Brownie (Truffle Suite, n.d.) (Brownie documentation, n.d.).

2.3 DISTRIBUTED DATA STORAGE

2.3.1 CENTRALIZED AND DECENTRALIZED CLOUD STORAGE SYSTEMS

Cloud Database-as-a-Service solutions utilize a global chain of servers that are accessible over the open Internet. Utilizing cloud storage services enables users to outsource, store and then remotely access their data on several different devices, providing users with an efficient and flexible way to manage their data (Cloud definition, n.d.). The trade-off for not needing to maintain a local data store is that the cloud storage service providers maintain and control all the client nodes and their data (MongoDB Atlas, n.d.). Notable examples are MongoDB Atlas (MongoDB Atlas, n.d.) and Microsoft Azure SQL Database (Databases on Azure, n.d.).

2.3.2 DECENTRALIZED CLOUD STORAGE SYSTEMS

Peer-to-peer networks enable individual computers to act as equally responsible nodes within the network to communicate directly to other nodes on the network and share information without the requirement of a central authority. The basic goals of P2P networks are to facilitate server-less data transfer, ad-hoc connectivity and rapid scalability between a network of untrusted peers (Hasan and Anwar, 2005).

BitTorrent is one of the most successful P2P file-sharing applications, optimizing resources by distributing pieces of data to clients and enabling them to swap the missing parts (Pouwelse, Garbacki, 2005) and has been implemented as a replacement to HTTP for file distribution for Web content in certain projects, most notably Zeronet (ZeroNet, n.d.).

2.3.3 BLOCKCHAIN-BASED DISTRIBUTED STORAGE SYSTEMS

The combination of blockchain technology and distributed storage systems enable an environment where the blockchain provides incentive and security for the files that are stored in the systems. There are a few popular blockchain-based distributed file systems including IPFS (IPFS Documentation, n.d.), Swarm (Swarm Distributed Storage, n.d.) and Storj (Storj Lab, 2018). IPFS is a P2P distributed file system for storing and accessing files, websites, applications and data; Swarm is an Ethereum based distributed storage platform and content distribution service; Storj is also a P2P decentralized cloud storage that enables data-sharing without a third party.

2.4 CURRENT CTI SHARING MODELS AND FRAMEWORKS

2.4.1 COLLABORATIVE THREAT INTEL

There is an initiative by Cyber Threat Alliance that aims to solve some of these key issues, such as the integration of STIX 2.0 to solve the issue of interoperability, contextualizing the MITRE ATT&CK framework. This initiative enabled collaboration between a few trusted organizations, including but not limited to, McAfee and Symantec, to form an intelligence-sharing model to improve their products and services to better protect their customers (Cyber Threat Alliance, n.d.).

CTA is a brilliant initiative that enables trusted peers to share private cyber threat intelligence using structured threat expression languages to solve the issue of interoperability. Each submission within CTA is assigned a point value, that is assigned to an entity, indicating their system supports pseudonymity alongside non-repudiation, which gives the member a value system. This idea incentivises legitimate uploads and enables further querying of the data, with who uploaded it, the threat actor name and the date of submission. The platform is cloud-based utilizing open-source tools to ensure a vendor-agnostic solution, using different membership programs to enable different levels of interaction with the system.

2.4.2 BLOCKCHAIN-BASED THREAT INTELLIGENCE SHARING

Recently, there has been more research into the problem domain of the trusted and secure methods of cyber threat information dissemination. Shiel et al. proposed a model that utilized the security properties intrinsic to blockchain technologies to overcome trust barriers and data privacy issues for CTI sharing. The project utilized MITRE's STIX 2.0 standard to structure threat data, as it is deemed to be the best path forward in threat dissemination. Through their analysis, they found that using a public blockchain network to share threat intelligence is not suitable due to the open nature. To overcome this issue, they used Hyper-ledger Fabric to enforce a private/permissioned distributed ledger and multiple levels of access control in the form of the permissioned nodes, traffic light protocols and partitioned Fabric channels to determine who should be able to see what data (Homan, Shiel and Thorpe, 2019). This focuses mainly on improving the integrity, interoperability and confidentiality of sharing private cyber threat intelligence to other incidence response teams, while also being able to communicate with the overall network as well.

As many CTI feeds that provide CTI services rely on open-source intelligence as a data collection channel, they can be left open to attackers injecting maliciously generated or modified data to compromise node reputation. Gong and Lee researched the complex and increasingly diverse attack vectors with the focus on attackers maliciously injecting data into a CTI platform. Utilizing the Ethereum blockchain and a permissionless Ganache test network, they proposed a framework that collects CTI from numerous sources, evaluates the validity of data and contributors, effectively distinguishing malicious contributors (Gong and Lee, 2020). The work explored different methods to increase confidence levels in data sources, improving the data integrity and eliminating inaccurate data, leading to Sybil resistance through the use of blockchain smart contracts. This focused on three contracts, one for user management, one for data reporting and one to disseminate threat-related alerts to consumers.

Purohit et al explored the ideas and concepts around leveraging blockchain technology to provide a threat intelligence sharing platform to defend against incoming cyber attacks. The solution makes use of a novel reputation system that use a set of protocols to rate peers objectively. The area within cyber threat intelligence sharing that they are focusing on is establishing distributed trust through the integration of reputation systems and automated control. The design makes use of both on-chain and off-chain components for storage, processing and sharing of threat intelligence and had a key focus on improving the reputation of the users interacting with the system (Purohit and Calyam, 2020).

3 METHODOLOGY

When developing software-based artefacts, planning is required to ensure that the product being delivered is completed on time and to a high standard. Many different development frameworks define best working practices throughout the project lifecycle. As most of these methodologies are focused on team-based working, I will apply my chosen methodology to a singular developer and will take on the role of the client in regards to testing.

3.1.1 WATERFALL

A waterfall is a structured approach that requires the project team to fully understand all elements of the project before it begins. The requirements need to be clearly defined, and the scope should be simple to be understood by all. Specialized tasks will be completed, reviewed and verified before moving onto the next phase. The waterfall is most effective when the developer cannot

often contact the end-user or client to sign off on any potential changes as the developers do not have any say in the analysis of requirements and planning (Delos Santos, 2020a).

3.1.2 AGILE

Agile methodology is an unstructured incremental development framework that focuses on results and a more flexible approach for the project. Rather than having the entire project scope from the start, the project development is broken down into smaller increments (Delos Santos, 2020b). As testing is done at every iteration, and developers generally work closely with testers errors can be fixed throughout the entirety of the development lifecycle. There are many different agile-based frameworks that all bring different advantages to the development of a project depending on certain factors.

3.1.2 SCRUM

Scrum is an agile methodology highly focused on team member function in a dynamic environment. This method requires team members to organize regular scrum meetings to keep track of the progress and decide where to go next. Scrum is broken down into events and artefacts. Sprints, which are fixed-length events where all the work required to reach the product goal occurs. This work includes sprint planning, which lays out all the work to be done for the sprint, daily scrums to review the progress and sprint reviews which looks at the outcome of the sprint and determines future changes. Scrum utilizes a product backlog, which is a prioritised list of features and alterations maintained by the product owner, as well as a sprint backlog, which specified the set of product backlog items selected for the sprint (What is Scrum? n.d.).

3.1.3 RAPID APPLICATION DEPLOYMENT

RAD is an agile methodology that starts with a loose definition of the requirements. The planning stage is brief as RAD puts a focus on the prototype iterations, but is core for the success of a project. With the initial scope, rapid production on prototyping certain features happens that can be demonstrated to the client for quick feedback. The rapid nature means that the project is more likely to have errors being caught much earlier in the development phase. Rapid construction then happens, which converts the prototypical features into a working model, receive feedback and it is entirely positive, move onto the final step, otherwise, resume prototyping. The final step is finalisation, where full-scale testing can happen (Rapid Application Development, 2021). It is suited for projects which can be implemented modularly and allow for more productivity with fewer developers (Rapid Application Development, 2021).

3.2 SOFTWARE DEVELOPMENT METHODOLOGY CHOICE

Rapid application deployment will be most suited for this project as it is highly suited for continuous and rapid development for a solo developer. Focusing on each stage individually allows full attention on certain aspects and incremental redesigns of the system which will be inevitable due to the concurrent research that is happening alongside it. The RAD methodology requires constant testing after each development phase, to gather feedback on where to improve it and the bugs identified, to ensure that each function works as intended.

This methodology will loosely utilize a Gantt Chart to structure the initial definition of requirements to follow a general order that makes sense for the production of the artefact (See Appendix A, Section 6) The order of priority for the requirements will be decided using MoSCoW (Must Have, Should-Haves, Could-Have-Wish) prioritization to be able to clearly define which functionalities should be focused on more heavily and the order in which to do it (MoSCoW Prioritization, n.d.).

4 SPECIFICATION

4.1 COMPARATIVE ANALYSIS OF TECHNOLOGICAL FEATURES

To determine the best tools and technologies that are most appropriate for the core objectives of this project, technical analysis of potential alternatives will be performed based upon the findings of the background and literature review.

4.1.1 CYBER THREAT INTELLIGENCE STANDARDS

COMPARISON OF CTI SHARING STANDARDS

	Identity	Motivation	Goal	Strategy	TTP	Tool	IOC	Atomic Indicators	Target	COA
OPENIOC					*	*	*	*		
STIX 1.0		*	*	*	*	*	*	*	*	*
STIX 2.0	*	*	*	*	*	*	*	*	*	*

Numerous ontologies intend to solve the issue of interoperability within the Cyber Threat Intelligence landscape, by utilizing a homogenous structured format to analyse intelligence formatted enabling a wider view of the entire environment. In this analysis, there was a focus on the enterprise-directed solutions, rather than the more specific frameworks that are designed for particular use-cases, such as NIST's SCAP.

FireEye's OpenIOC XML schema is a very basic schema that shares characteristics that identify a known threat, attacker methodology or other indicators of compromise. The information covered by OpenIOC is mainly derived from low-level atomic indicators, covering only a small section of the CTI landscape. For this project, a wider scope of threat intelligence is required to give a better understanding of the current landscape. The best choice for enterprise-suitable threat intelligence sharing standards would be MITRE’s STIX, which encases not only indicators of compromise but covers a wider view of threat information.

STIX 1.X utilized an XML Schema that utilized a wide range of core components (Appendix D) and leveraged elements from other schemas, most notably CybOX, to best represent threat information (Appendix E). STIX 2.X merges CybOX and STIX 1.X into one, with CybOX objects now being referred to as Cyber Observables. STIX 2.X utilizes JSON serialization rather than XML, and although there are benefits to both, JSON is more lightweight, simpler to use and is preferred by many developers. The incorporation of JSON enables features for clear semantic representation of Cyber Threat Intelligence (Cyber Threat Intelligence Technical Committee, n.d.). STIX also utilizes the TAXII protocol, which enables intelligence exchange over HTTPS (Oasis-Open, n.d).

4.1.2 BLOCKCHAIN PERMISSIONS

COMPARISON BETWEEN BLOCKCHAIN CLASSIFICATIONS

	Consensus Mechanism	Identity	Protocol Efficiency	Immutability	Ownership & Management	Transaction Approval
PUBLIC	- Costly Proof-Of-Work - Decentralized (all miners)	- Potentially malicious - Pseudo-anonymous	- Low efficiency - High energy	Highly immutable	-Public -Permissionless	Minutes
PRIVATE	- Light Proof-Of-Work - Custom Consensus - Centralised Organization	- Trusted identified users	- High efficiency - Low energy	Susceptible to collusion attacks	-Centralised -Permissioned whitelist	Milliseconds
FEDERATED	- Proof-Of-Work - Custom Consensus - Leader nodes set	- Trusted identified users	- High efficiency - Low energy	- Susceptible to collusion attacks	-Semi - Centralised -Permissioned nodes	Milliseconds

Although public blockchains are highly immutable and require no permissions to join the network, they are limited in their custom-ability and slow transaction times (Casino and Patsakis, 2019). Ethereum is an example of a public, smart contract enabled blockchain. The cost of gas to perform transactions on the Ethereum network is rising and this is due to the high volume of trading and activity occurring on the blockchain; The high gas fees will be eradicated when Ethereum 2.0, utilizing Proof-Of-Stake consensus, is deployed(Ethereum 2.0 Gas Issues and Scale, 2020).

Both private and federated blockchains, which are permissioned networks, allow for a much higher level of customisation, being able to set transactions fees for the network, a higher potential for scalability and even the use of consensus protocols such as Proof of Elapsed Time (PoET), Raft and Istanbul BFT, which are not possible on public blockchains (Dyatko, 2019). It mainly comes down to the fact that permissionless blockchains tend to focus more on business to consumer scenarios, whereas permissioned blockchains are structured well for business to business scenarios (Massessi, 2018).

Federated blockchain networks take elements from both private and public, and rather than granting a single entity control, it is granted to a group of approved organizations. Federated blockchains, therefore, take certain security elements from public blockchains, while also enabling a significant degree of control of the network. This allows for higher levels of customisability in regards to interaction on the network, where the API may be open to the public, but is restrictive in what actions can be performed by external entities. Examples of federated blockchains include Quorum, Hyper-ledger and Corda (Shkoor, 2019).

4.1.3 FEDERATED BLOCKCHAIN TECHNOLOGIES

COMPARISON BETWEEN HYPER-LEDGER, CORDA AND QUORUM

	Node Permissions	Identity	Consensus Algorithm	Smart Contract Logic	Smart Contract Languages
HYPER-LEDGER FABRIC	Configurable: • Node • Channel • Consortium levels	PKI with native organization ID	KAFKA RAFT	ChainCode	• JavaScript • Go • Java
CORDA	File based configuration on each node	PKI with personal and organization ID	Validated by Notaries	Ricardian Contracts	• Kotlin • Java
ETHEREUM QUORUM	Smart contract permission models	Node ID uses public/private key	• Clique • RAFT • IBFT	• Public Contracts • Private Contracts	• Solidity

Enterprise-ready blockchains are popular and can all offer privacy, robust security mechanisms, governance and adaptable characteristics (GoQuorum, n.d.). For this project, Ethereum's Quorum will be used as the blockchain network as there are a lot of tools to help develop distributed applications for Ethereum based projects; Quorum utilizes Proof of Authority consensus mechanisms, most notably RAFT, which offers consensus via an elected leader, who will replicate the log to all the other nodes within the network (Raft Consensus Algorithm, n.d.).

4.1.4 BACK-END PROGRAMMING LANGUAGES & DEVELOPMENT FRAMEWORKS

When it comes to programming the back-end of Ethereum-based distributed applications there are two key programming languages to consider: JavaScript and Python. Both of these languages have their Ethereum APIs to interact with the blockchain, web3.js and web3.py (web3js, n.d.) (web3py, n.d.). They both also have extensive development frameworks, JavaScript has Truffle and Python has Brownie, which assists in the development and deployment of smart contracts to the EVM (Truffle Suite, n.d.)(Brownie documentation, n.d.).

JavaScript is considered the de-facto front-end language, so the community around web3.js development is larger and there is much more support with tools and tutorials. Although the reading materials around Python blockchain development were lesser, Python is deemed the most appropriate language to use for this project as the STIX Cyber Threat Intelligence library has been designed and created in Python, making integration into the core functional aspect of the code flawless.

4.1.5 DISTRIBUTED STORAGE SYSTEMS

Traditional data storage systems are often organized in relational databases using SQL, and utilize a static schema. There is a heavy focus on ACID compliance for traditional storage systems, which looks at atomicity, consistency, isolation and durability. Microsoft's Azure SQL DB is a traditional-cloud based storage system (Databases on Azure, n.d.).

Document structured storage systems allow for much higher customizability, utilizing a dynamic schema to store the data. This is an example of a NoSQL database, which are non-tabular and store data differently to relational databases. An example of this is MongoDB, which is a document database that used to store JSON-like documents. MongoDB is highly suited for the type of data storage required for this project, evidenced by the TAXII protocol that is designed to transfer STIX requires the use of a MongoDB. MongoDB Cloud Atlas is a cloud database that enables document structured, server-less storage (MongoDB Atlas, n.d.).

DISTRIBUTED STORAGE SYSTEMS TECHNICAL COMPARISON

	DECENTRALIZED	DATA CONFIDENTIALITY	DATA INTEGRITY	SPEED	DYNAMIC FORMAT
AZURE SQL DB	No	Yes	No	Fast	No
MONGODB ATLAS	No	Yes	No	Fast	Yes
ETHEREUM SWARM	Yes	No	Yes	Slow	Yes
IPFS	Yes	No	Yes	Slow	Yes

CLOUD STORAGE

The use of cloud storage can be very useful in certain scenarios, allowing third-party auditor on behalf of the cloud client to verify the integrity of the data stored in the cloud. The centrality of cloud storage solutions means that generally, the data can be maintained better by the central authority. Some key problems are inherent to a cloud-based storage solution that make it not suited for this kind of project. Cloud-hosted storage systems tend to be prone to system overloads and targetted attacks, including DDoS, APTs and Crypto-Jacking (Purohit and Calyam, 2020). The centrality that comes with cloud storage solutions is not suited to data that requires high availability and high integrity; This is because users do not physically own their data after outsourcing it to cloud storage providers, so the matter of whether the data remains intact on the cloud servers or not will always be unknown, thus the data cannot have true integrity (Zhang and Xu, 2017).

PEER-TO-PEER STORAGE

Rather than relying on a centralised cloud solution for the storage, data can be stored in decentralized P2P storage solutions. Distributed document storage has many advantages over traditional data storage, including low-latency retrieval, a reliable and fault-tolerant operation providing intermittent availability through redundant storage and zero-downtime. Both Swarm and IPFS are solutions that utilize a BitTorrent like P2P protocol to store data in a decentralized and distributed manner over a large number of nodes.

Swarm and IPFS are both comprehensive decentralized storage solutions for the next generation of the internet, providing integrity through content addressing, with the high-level goals and

technologies used being very similar. Both Swarm and IPFS are suitable solutions for this project, but IPFS will be used as it has been around longer, meaning there is more support and development assistance around IPFS and the main differences between the two storage systems aren't going to affect the outcome of the artefact, as they will both provide an integrity-driven storage system with high availability (Cyber Threat Intelligence Technical Committee, n.d.).

4.2 FEATURE SPECIFICATION

The requirements for the artefact were derived from both the literature review and the comparative analysis of different technologies that are required to complete a successful prototype.

ID #	REQUIREMENT	MoSCoW
1	A federated blockchain will dictate the pre-defined permissioned nodes that will have access to a distributed application.	Must have
2	User must be able to convert unstructured threat information into STIX expressions (Indicator, Tool, Attack Pattern, Malware)	Must have
3	User must be able to convert unstructured threat information into STIX expressions (Vulnerability, Course of Action, Threat Actor, Infrastructure, Intrusion Set)	Should have
4	User must be able to convert unstructured threat information into STIX expressions (Campaign, Location, Observed Data, Grouping, Malware Analysis, Note, Opinion, Report, Sighting)	Could have
5	The application should verify whether the data is in a valid format and is compliant with the STIX language	Must Have
6	The application should upload a JSON-serialized STIX object on a decentralized and distributed peer-to-peer data storage system	Must Have
7	Each STIX object uploaded to the peer-to-peer data storage should be done consecutively with the smart contracts, linking the distributed storage ID to the STIX object so it can later be queried.	Must Have
8	The application should allow users to add relationships linking STIX objects	Should Have
9	The application will decide the relationship type based on the source reference type and the target type	Should Have
10	The user should be able to retrieve data from the peer-to-peer data storage by querying key indexed values	Must Have
11	User must be able to query STIX Indicators using their pattern (File Hash, IPV4 Address, URL Address)	Must Have
12	User must be able to query all STIX Objects, excluding Indicators, by searching the name of the object.	Should Have
13	User can find all other objects that have a relationship with the queried one	Could Have
14	The application will be encased in a front-end user interface	Wish
15	Users external to the system will be able to query for STIX threat data	Wish

5 DESIGN

5.1 INITIAL SOLUTION DIAGRAM

At the start of the project, the idea was to utilize all aspects of the STIX protocol, which included TAXII, the application layer protocol to assist in the communication of cyber-threat intelligence using the TAXII Python libraries: cti-taxii-server and cti-taxii-client (Oasis-Open, n.d) The TAXII server included a back-end plug-in for MongoDB. The library included customizability with the back-end plug-ins, meaning that the artefact would have been able to utilize a distributed data storage solution such as MongoDB Atlas, but the data would have been owned by a third party cloud service provider and not the users on the chain. The prototype design for the initial solution was set up with a local Mongo database, and during integration to the cloud solution, the issues of centrality became apparent and were deemed inappropriate (See Appendix F). The initial system prototype design incorporated many choices that were altered as the research went on to provide the best core system.

5.2 SECOND SOLUTION DIAGRAM

As the TAXII/MongoDB Atlas structure did not decentralized and the users of the blockchain network were using a permissioned blockchain, there was an exploration into the idea of utilizing the use of events within the Solidity language to store the small JSON-like files within the Ethereum transaction logs to ensure a more efficient data storage (See Appendix G). The events within solidity have three main use cases, which include smart contract return values for the user interface, asynchronous data triggers and a cheaper form of storage (Introduction to Events and Logs in Ethereum, 2016). The idea of emitting events that contained the STIX data was explored and prototyped, but ultimately deemed infeasible for this project as there is no way to be able to add STIX relationship objects, and any future works would be highly limited in how new features are incorporated.

5.3 FINAL SOLUTION DIAGRAM

The final design utilizes some of the same core components as solution one and two while incorporating a more efficient and integrity-driven distributed storage solution, IPFS, to store the threat intelligence data (See Figure 2 below).

5.3.1 BLOCKCHAIN NETWORK

Quorum was considered the best option throughout the entirety of the project. The ability to create a semi-centralized consortium blockchain utilizing all the same tools that come with developing standard Ethereum dApps allows for more creative freedom in prototype creation. The Quorum network also enables the usage of the Raft vote-based consensus protocol, which allows for fast transaction times due to the leader-follower model, which is more energy and time efficient than traditional consensus protocols such as PoW. Four nodes were chosen to simulate each participating organization in this design as this is only a prototype and not yet custom-built.

5.3.2 SMART CONTRACTS

As seen in Appendix F, the initial prototype utilised one contract to standardise the threat intelligence, one to upload the threat intelligence and another to query the threat intelligence. Through the RAD development methodology, the rapid creation of function prototypes and testing showed that the best method was to utilize the smart contracts as an aspect of the immutable storage solution, rather than the core application functionality. Each contract will contain a key-value mapping store, which will allow the required data value to be retrieved based on the key.

5.3.3 BACKEND FUNCTIONS

The first prototype utilized JavaScript for the backend of the application. This was not optimal when the structured threat expression package is in Python pip and not a JavaScript node module and was realised early in the development. The packages to be used with the final solution will be web3.py to interact with the Quorum Blockchain, stix2.py to convert unstructured threat information into structured threat expressions, and ipfshttpclient to interact with the distributed nodes in the IPFS network.

5.3.4 DATA STORAGE

After the second design attempt, the idea of storing the data in a decentralized and distributed form became one of the most integral to the project. The initial plan for the blockchain was to keep a log of when the data had been uploaded to the cloud storage service to ensure integrity in the system, but as development and research went on, it no longer made any sense that the consortium of members should rely on third-party service providers when the data integrity and availability are paramount. The second prototype attempted to incorporate a feature from the Ethereum ecosystem that was not built for object data storage with unique values that can be queried. Instead, the research looked into decentralized and distributed storage systems. This means that the data will be accessible at all times, and the use of content addressing means that the data can not be tampered with, as the location is based upon the contents of the data.

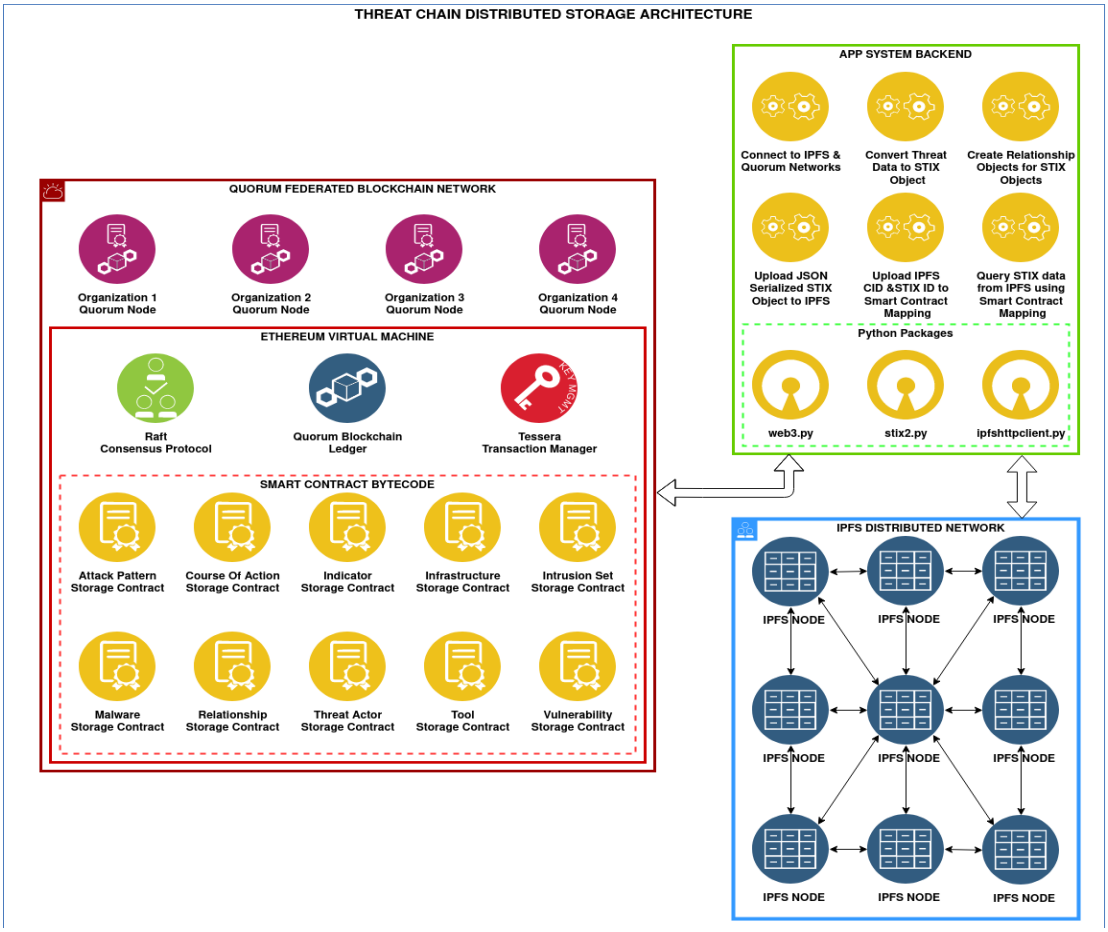


Figure 2: Final Solution

6 IMPLEMENTATION

Using the Rapid Application Development to quickly develop prototypes allowed for complete flexibility in the approach taken. This enabled development to be done with a holistic view of how each section intertwines and works cohesively together, being able to go back and completely change how certain sections functioned. Many tools and techniques were explored, tested and ultimately decided against for potentially slightly inferior tools and techniques on the lower level, but when combined with the rest of the system, enable the possibility for a fully distributed federated threat intelligence platform a system that with regards to the integrity and availability, is superior to current solutions relying on in-house centralised servers or external third-party providers. The RAD methodology for development, discussed in Section 3, requires a loose set of requirements, that were initially proposed in the introduction (Section 1) and then further refined in the specification (Section 4), which served to create the development stages for this project.

6.1 RAPID APPLICATION DEVELOPMENT CYCLE

DEVELOPMENT PHASE	REQUIREMENT ID	PROGRESSION
0	-	Set up tools, systems and environments
1	1	Creation of a permissioned blockchain system
2	2, 3	Creation of multiple forms to input unstructured CTI: Indicator, Tool, Attack Pattern, Malware, Vulnerability, Course Of Action, Threat Actor, Infrastructure, Intrusion Set
3	2, 3, 5	Conversion of User Input to STIX Object
4	6	Preperation and Upload of JSON-STIX Objects to IPFS
5	7,8	Creation of smart contracts to store : • STIX ID • STIX Object Name • IPFS Content ID
6	10, 11	Creation of STIX Indicators query function
7	10, 12	Creation of STIX Objects (Excluding Indicators) query function
8	8	Creation of form to manually input STIX Relationship Object
9	9	Creation of function that choses relationship type dependant on input values
10	13	Creation of function to retrieve all related STIX objects

6.2 DEVELOPMENT PHASE 0: TOOLS, SYSTEMS AND LANGUAGES USED

There were many tools that required an initial set up and installation on the local client to interact with with the application, including the GoQuorum client, IPFS client and the Remix-IDE.

6.2.1 QUORUM BLOCKCHAIN ENVIRONMENT

The Quorum Blockchain platform was created by modifying the Quorum 7 Node example that ConsenSys released on GitHub(Quorum Examples, 2016). The network can be run locally, using vagrant, which would create a virtual environment using VirtualBox, or docker, which creates containerized applications that utilizes a small configuration file. The best option for the artefact was to run it through a docker container, as it has a much quicker start-up and image build time than vagrant, since it is not running a resource-heavy virtual environment (Palmer, n.d.), which is suited better for the RAD cycle.

The docker configuration file was configured for the artefact. The number of permissioned nodes was decreased to 4, rather than the pre-defined 7 as to fit with the final system design in Section 5. The docker config sets the Istanbul BFT consensus mechanism as default, so when the environment was being initially run, the consensus mechanism was set to Raft as it is the optimal consensus for a federated blockchain. After the docker images have been initialised, the blockchain platform can be run using docker-compose, as seen in Appendix H.

6.2.2 IPFS DISTRIBUTED STORAGE ENVIRONMENT

To use an IPFS node, an IPFS client needed to be installed to allow the local system to communicate with the IPFS network. The GoIPFS API package was installed locally, which treats the local system as an IPFS node. The IPFS API is run using a daemon running in the background ready to receive communication from the application (See Appendix I).

6.2.3 BACKEND DEVELOPMENT ENVIRONMENT

VSCode is the IDE that was used to develop the artefact. The Python programming language was chosen for the project, so a development environment was set up with relevant packages installed. This was done by using PipEnv, which is a python package manager that was used to install the Python packages to a virtual environment by using a PipFile (See Figure 3). The PipEnv can be run in a command-line shell, enabling easy interaction with Python modules.

```
flash@flash-ThinkPad-X1-Carbon-2nd:~/threat_chain/project/interface$ cat Pipfile
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]

[packages]
flask = "*"
web3 = "*"
stix2 = "*"
ipfshttpclient = "==0.7.0a1"

[requires]
python_version = "3.8"
```

Figure 3: Pipfile Python Packages

6.2.4 SMART CONTRACT DEVELOPMENT ENVIRONMENT

Solidity is the chosen smart contract language to use for the Quorum blockchain artefact. Some of the smart contract development was completed within the VSCode IDE, but the majority of the work was done within the Remix Web Browser IDE. Remix is a web-based IDE that allows the development, deployment and administration of smart contracts for Ethereum-like blockchains (Remix IDE, n.d.). Using the Remix-IDE API, Remix can connect to the local systems file storage environment (See Figure 4), which made the incorporation of the contracts much easier. Remix will compile the code, and will then deploy it to an active Web3 environment, allowing for fast smart contract deployment to the system (See Appendix J). The alternative smart contract development frameworks that were tried out were Truffle (JavaScript) and Brownie (Python), which were designed to use Ganache, a personal blockchain testnet. These frameworks were not suitable as they would not be able to connect to the custom blockchain, so Remix was utilized as the smart contract development, testing and deployment tool.

```
flash@flash-ThinkPad-X1-Carbon-2nd:~/threat_chain/project/interface/backend$
remixd -s sol/ --remix-ide http://remix.ethereum.org/#optimize=false&runs=200
&evmVersion=null&version=soljson-v0.8.1+commit.df193b15.js
[4] 459629
[5] 459630
[6] 459631
[3]   Done                    evmVersion=null
[5]   Done                    runs=200
```

Figure 4: Connect Remix Web IDE to Local Storage

6.3 DEVELOPMENT PHASE 1: CREATION OF PERMISSIONED BLOCKCHAIN SYSTEM

6.3.1 CONNECTION INITIALISATION

Phase 1 of project development was to create a basic function that enables the 4 permissioned Quorum nodes to authenticate their identity to prove that they are part of the federated blockchain. This method was not intended to be a secure method, but just a simulation of what a fully developed, test-hardened solution would provide. When user credentials are verified to be correct, a connection to the blockchain node is made, the associated blockchain address is retrieved and the system connects to the IPFS network. The basic authentication checks will only allow valid accounts can move onto the main menu, as seen in Figure 5.

```
Threat-Chain CLI Application

enter org name: TCN1
enter password: pass1

Welcome TCN1 , your address is 0xed9d02e382b34818e88B88a309c7fe71E65f419d

1: upload threat intelligence object
2: retrieve threat intelligence object
enter:
```

Figure 5: Authentication and Menu

```
1: upload threat intelligence object
2: retrieve threat intelligence object
enter: 1

threat intelligence objects:
1 : indicator
2 : malware
3 : tool
4 : vulnerability
5 : attack pattern
6 : course of action
7 : threat actor
8 : infrastructure
9 : intrusion set
0: relationship
select 0 - 9:
```

Figure 6: Upload Menu Options

6.4 DEVELOPMENT PHASE 2: CREATION OF UNSTRUCTURED CTI UPLOAD FORMS

Phase 2 of the project development looked into creating the functions and user inputs that were required to further process the data and create the relevant STIX object classes. All the STIX object attributes were sourced from the STIX CTI documentation (Cyber Threat Intelligence Technical Committee, n.d.). Each choice can be selected from a menu by the user (See Figure 6) to create a new object (See Figures 7 – 15). The values in the relevant figures were set to either 1 or “” to present a generic version form.

6.4.1 ATTACK PATTERN UPLOAD FORM

```
attack pattern name:
attack pattern description:

enter external reference:

enter source name:
enter description:

enter url y/n: n

enter id y/n: n

enter number of aliases: 1
enter alias:

number of killchain phases: 1

lockheed martin killchain phases:
1: reconnaissance
2: weaponization
3: delivery
4: exploitation
5: installation
6: command and control (C2)
7: actions on objectives
kill-chain phase 1-7: 1
```

Figure 7: Attack Pattern Form

6.4.2 INDICATOR UPLOAD FORM

```
indicator name:
indicator description:

indicator types:
1: anomalous-activity
2: anonymization
3: benign
4: compromised
5: malicious-activity
6: attribution
indicator type 1-7 : 1

pattern types:
1: file
2: ipv4
3: url
pattern type 1-3: 3

enter url:
```

Figure 8: Indicator Form

6.4.3 INFRASTRUCTURE UPLOAD FORM

```
infrastructure name:
infrastructure description:

infrastructure types:
1: amplification
2: anonymization
3: botnet
4: command-and-control
5: exfiltration
6: hosting-malware
7: hosting-target-lists
8: phishing
9: reconnaissance
10: staging
11: unknown
infrastructure type 1-11 : 1

enter number of aliases: 1
enter alias:

number of killchain phases: 1

lockheed martin killchain phases:
1: reconnaissance
2: weaponization
3: delivery
4: exploitation
5: installation
6: command and control (C2)
7: actions on objectives
kill-chain phase 1-7: 1
```

Figure 9: Infrastructure Form

6.4.3 INTRUSION SET UPLOAD FORM

```
intrusion set name:
intrusion set description:

enter number of aliases: 1
enter alias:

number of goals: 1
enter goal:

resource levels:
1: individual
2: club
3: contest
4: team
5: organization
6: government

resource levels 1-6: 1

attacker motivations:
1: accidental
2: coercion
3: dominance
4: ideology
5: notoriety
6: organizational-gain
7: personal-gain
8: personal-satisfaction
9: revenge
10: unpredictable
```

Figure 10: Intrusion Set Form

6.4.5 MALWARE UPLOAD FORM

```
malware name:
malware description:

malware types:
1: adware
2: backdoor
3: bot
4: bootkit
5: ddos
6: downloader
7: dropper
8: exploit-kit
9: keylogger
10: ransomware
11: remote-access-trojan
12: resource-exploitation
13: rogue-security-software
14: rootkit
15: screen-capture
16: spyware
17: trojan
18: unknown
19: virus
20: webshell
21: wiper
22: worm
malware type 1-22: 1

1: malware family
2: malware instance
family or instance: 1

number of killchain phases: 1

lockheed martin killchain phases:
1: reconnaissance
2: weaponization
3: delivery
4: exploitation
5: installation
6: command and control (C2)
7: actions on objectives
kill-chain phase 1-7: 1
```

Figure 11A: Malware Form

```
number of malware capabilities: 1
malware capabilities:
1: accesses-remote-machines
2: anti-debugging
3: anti-disassembly
4: anti-emulation
5: anti-memory-forensics
6: anti-sandbox
7: anti-vm
8: captures-input-peripherals
9: captures-output-peripherals
10: captures-system-state-data
11: cleans-traces-of-infection
12: commits-fraud
13: communicates-with-c2
14: compromises-data-availability
15: compromises-data-integrity
16: compromises-system-availability
17: controls-local-machine
18: degrades-security-software
19: degrades-system-updates
20: determines-c2-server
21: emails-spam
22: escalates-privileges
23: evades-av
24: exfiltrates-data
25: fingerprints-host
26: hides-artifacts
27: hides-executing-code
28: infects-files
29: infects-remote-machines
30: installs-other-components
31: persists-after-system-reboot
32: prevents-artifact-access
33: prevents-artifact-deletion
34: probes-network-environment
35: self-modifies
36: steals-authentication-credentials
37: violates-system-operational-integrity

malware capability 1-37: 1
```

Figure 11B: Malware Form

6.4.6 COURSE OF ACTION UPLOAD FORM

```
course of action name:
course of action description:
```

Figure 12: Course Of Action Form

6.4.7 THREAT ACTOR UPLOAD FORM

```
threat actor name:
threat actor description:

threat actor types:
1: activist
2: competitor
3: crime-syndicate
4: criminal
5: hacker
6: insider-accidental
7: insider-disgruntled
8: nation-state
9: sensationalist
10: spy
11: terrorist
12: unknown

number of threat actor types: 1
threat actor type 1-12 : 1

enter number of aliases: 1
enter alias:

threat actor roles:
1: agent
2: director
3: independant
4: infrastructure-architect
5: infrastructure-operator
6: malware-author
7: sponser

number of threat actor roles: 1
threat actor role 1-7 : 1

number of goals: 1
enter goal: 1
```

Figure 13A: Threat Actor Form

```
threat actor sophistication:
1: none
2: minimal
3: intermediate
4: advanced
5: expert
6: innovator
7: strategic

sophistication levels 1-6: 1

resource levels:
1: individual
2: club
3: contest
4: team
5: organization
6: government

resource levels 1-6: 1

attacker motivations:
1: accidental
2: coercion
3: dominance
4: ideology
5: notoriety
6: organizational-gain
7: personal-gain
8: personal-satisfaction
9: revenge
10: unpredictable

attacker motivations 1-10: 1
```

Figure 13B: Threat Actor Form

6.4.8 TOOL UPLOAD FORM

```
tool name:
tool description:

tool types:
1: denial-of-service
2: exploitation
3: information-gathering
4: network-capture
5: credential-exploitation
6: remote-access
7: vulnerability-scanning
8: unknown
enter number of tool types: 1
enter tool type: 1

enter number of aliases: 1
enter alias:

number of killchain phases: 1

lockheed martin killchain phases:
1: reconnaissance
2: weaponization
3: delivery
4: exploitation
5: installation
6: command and control (C2)
7: actions on objectives
kill-chain phase 1-7: 1
enter tool version: 1
```

Figure 14: Tool Form

6.4.9 VULNERABILITY UPLOAD FORM

```
vulnerability name:
vulnerability description:

enter external reference:

enter source name:
enter description:

enter url y/n: n

enter id y/n: n
```

Figure 15: Vulnerability Form

6.5 DEVELOPMENT PHASE 3: CONVERT UNSTRUCTURED CTI INTO STIX OBJECT

One of the core components of the system is the interoperability that comes from using the STIX patterning language. The STIX package utilizes custom class objects for each STIX Object, requiring specialized functions to set some of the class attribute data structures. The upload forms for each STIX object have been integrated into the class definition, rather than doing it externally and then setting each class attribute to a variable (See Section 6.4) (See Figure 16).The object vocab module contains functions for setting different attribute values that may be unique to a particular object or common across many, which enabled efficient reusage of certain required pattern types. The general class object format that is created can be seen in Figures 17 – 25.

```python
import object_vocab as ov
from web3 import Web3
from stix2 import (AttackPattern, parse)

def uploadAttackPattern():
    #STIX ATTACK PATTERN CLASS OBJECT CREATION
    attack_patternSTIX = AttackPattern(
        name = input("\nattack pattern name: "),
        description=input("attack pattern description: "),
        external_references=ov.externalReference(),
        aliases=ov.aliases(),
        kill_chain_phases= ov.enumerateKCP()
    )
```

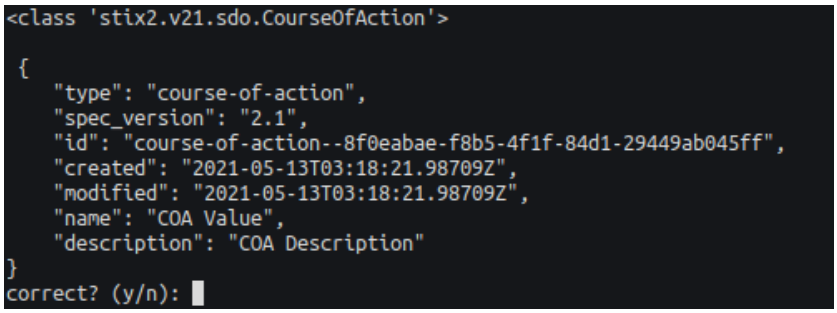
Figure 16: STIX Class Object Definition

6.5.1 ATTACK PATTERN CONVERSION

```
<class 'stix2.v21.sdo.AttackPattern'>

{
    "type": "attack-pattern",
    "spec_version": "2.1",
    "id": "attack-pattern--c06ad113-4d67-4241-abd6-2906ee440d9f",
    "created": "2021-05-13T03:28:09.405142Z",
    "modified": "2021-05-13T03:28:09.405142Z",
    "name": "ATTPTT Value",
    "description": "ATPT desc",
    "aliases": [
        "ATTPTT #1",
        "ATTPTT #2"
    ],
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "delivery"
        },
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "command and control (C2)"
        }
    ],
    "external_references": [
        {
            "source_name": "CAPEC",
            "description": "capec desc",
            "url": "('http://www.capec.com',)",
            "external_id": "CAPEC-123"
        }
    ]
}
```


Figure 17: Attack Pattern STIX Object

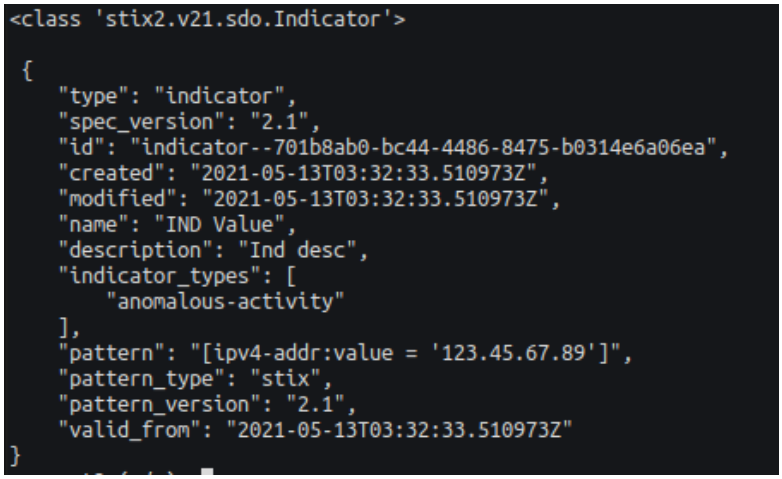
6.5.2 COURSE OF ACTION CONVERSION



```
<class 'stix2.v21.sdo.CourseOfAction'>

{
    "type": "course-of-action",
    "spec_version": "2.1",
    "id": "course-of-action--8f0eabae-f8b5-4f1f-84d1-29449ab045ff",
    "created": "2021-05-13T03:18:21.98709Z",
    "modified": "2021-05-13T03:18:21.98709Z",
    "name": "COA Value",
    "description": "COA Description"
}
correct? (y/n):
```


Figure 18: Course Of Action STIX Object

6.5.3 INDICATOR CONVERSION



```
<class 'stix2.v21.sdo.Indicator'>

{
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--701b8ab0-bc44-4486-8475-b0314e6a06ea",
    "created": "2021-05-13T03:32:33.510973Z",
    "modified": "2021-05-13T03:32:33.510973Z",
    "name": "IND Value",
    "description": "Ind desc",
    "indicator_types": [
        "anomalous-activity"
    ],
    "pattern": "[ipv4-addr:value = '123.45.67.89']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T03:32:33.510973Z"
}
```


Figure 19: Indicator STIX Object

6.5.4 INFRASTRUCTURE CONVERSION

```
<class 'stix2.v21.sdo.Infrastructure'>

{
    "type": "infrastructure",
    "spec_version": "2.1",
    "id": "infrastructure--73c5dc81-0ac8-4a2d-8476-79691f056681",
    "created": "2021-05-13T03:34:55.154731Z",
    "modified": "2021-05-13T03:34:55.154731Z",
    "name": "INF Value",
    "description": "inf desc",
    "infrastructure_types": [
        "botnet"
    ],
    "aliases": [
        "INF Alt"
    ],
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "delivery"
        },
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "command and control (C2)"
        }
    ]
}
```

Figure 20: Infrastructure STIX Object

6.5.5 INTRUSION SET CONVERSION

```
<class 'stix2.v21.sdo.IntrusionSet'>

{
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--c64372bb-d4a4-437b-9501-8ad533a00a02",
    "created": "2021-05-13T03:41:16.617451Z",
    "modified": "2021-05-13T03:41:16.617451Z",
    "name": "IS Value",
    "description": "is desc",
    "aliases": [
        "IS Value Alt"
    ],
    "goals": [
        "Goal #1"
    ],
    "resource_level": "club",
    "primary_motivation": "coercion"
}
```

Figure 21: Intrusion Set STIX Object

6.5.6 MALWARE CONVERSION

```
<class 'stix2.v21.sdo.Malware'>

{
    "type": "malware",
    "spec_version": "2.1",
    "id": "malware--5cea3cd9-8ec4-4d0d-a731-c9e081d0a97e",
    "created": "2021-05-13T03:45:03.42915Z",
    "modified": "2021-05-13T03:45:03.42915Z",
    "name": "MAL Value",
    "description": "Mal desc",
    "malware_types": [
        "adware"
    ],
    "is_family": true,
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "reconnaissance"
        },
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "delivery"
        }
    ],
    "capabilities": [
        "anti-vm",
        "fingerprints-host"
    ]
}
```

Figure 22: Malware STIX Object

6.5.7 THREAT ACTOR CONVERSION

```
<class 'stix2.v21.sdo.ThreatActor'>

{
    "type": "threat-actor",
    "spec_version": "2.1",
    "id": "threat-actor--0fa4dc78-d780-46e4-be24-412d906f37ab",
    "created": "2021-05-13T03:47:07.190115Z",
    "modified": "2021-05-13T03:47:07.190115Z",
    "name": "TA Value",
    "description": "TA desc",
    "threat_actor_types": [
        "activist"
    ],
    "aliases": [
        "TA Alt"
    ],
    "roles": [
        "director",
        "sponser"
    ],
    "goals": [
        "Goal Val"
    ],
    "sophistication": "minimal",
    "resource_level": "individual",
    "primary_motivation": "accidental"
}
```

Figure 23: Threat Actor STIX Object

6.5.8 TOOL CONVERSION

```
<class 'stix2.v21.sdo.Tool'>

[
    "type": "tool",
    "spec_version": "2.1",
    "id": "tool--4970bb1a-eab2-4361-9691-e41e3f707de9",
    "created": "2021-05-13T03:49:12.699547Z",
    "modified": "2021-05-13T03:49:12.699547Z",
    "name": "TOOL Val",
    "description": "tool desc",
    "tool_types": [
        "1"
    ],
    "aliases": [
        "TOOL Alt"
    ],
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "delivery"
        },
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "exploitation"
        }
    ],
    "tool_version": "2"
```

Figure 24: Tool STIX Object

6.5.9 VULNERABILITY CONVERSION

```
<class 'stix2.v21.sdo.Vulnerability'>

{
    "type": "vulnerability",
    "spec_version": "2.1",
    "id": "vulnerability--962cd9d8-0f71-47d9-9e9f-da7fba6217de",
    "created": "2021-05-13T00:05:30.864546Z",
    "modified": "2021-05-13T00:05:30.864546Z",
    "name": "",
    "description": "",
    "external_references": [
        {
            "source_name": "",
            "description": ""
        }
    ]
}
```

Figure 25: Vulnerability STIX Object

6.6 DEVELOPMENT PHASE 4: PREPARE AND UPLOAD STIX-JSON OBJECT TO IPFS

Following the creation of the STIX CTI Object, the STIX Class Object is converted to a JSON-like string and loaded into a Python Dictionary. The Python IPFS Client enables writes JSON formatted strings to a storage location in the IPFS network based on its content (See Figure 26). The hash of location is then returned to a variable ready for further use and displayed to the user to show a valid upload (See Figure 27).

```
#confirmation that STIX Object is valid
confirm = input("correct? (y/n): ")
if (confirm.upper() == "Y"):
    #converts the STIX object into a JSON object
    indicator = json.loads(str(indicatorSTIX))
    #uploads json object to IPFS and records the content address
    contentAddr = ipfs_cli.add_json(indicator)
    print("ipfs content address: ", contentAddr)
```

Figure 26: Code snippet for IPFS data upload and CID store

```
correct? (y/n): y
ipfs content address:  Qmb7AsZueR4KY9nrshjXn29ZX9JpBVkYmasousadAE7qUg
block:  9

continue y/n:
```

Figure 27: CLI Output for IPFS upload

6.7 DEVELOPMENT PHASE 5: CREATION OF SMART CONTRACTS

The next phase in the development of the project artefact was to create the Solidity smart contracts to store an immutable copy of the IPFS location of the STIX object. To enable the completion of feature requirement #12, query data from the distributed storage (See Section 4), a data type called a struct is used. A struct is a custom type that can be defined, constising of a name and associated properties (See Figure 28). The artefact uses two structures: the first stores the STIX ID with the value to query the STIX object by; For indicator objects the value to query is the pattern attribute (IPV4, File Hash, URL) and for the rest of the objects it's the name attribute. The second structure stores the IPFS Content Address with the STIX ID.

To create an association between these values, a key-value pair data store was required. Where most other programming languages make use of a dictionary to store key-value pairs, Solidity uses a data structure called a Mapping. The first set of data mapping is the attribute value (key) to the first struct (value) containing the STIX ID and attribute, followed by the mapping of the STIX ID (key) to the IPFS content address (value) (See Figure 29). This is the final stage in the conversion and upload of CTI, in which the current block number will be returned and an option to go back to main menu appears.

```
struct AttackPatternID{
    string stixID;
    string att_pat_val;
}
struct DataLocation{
    string contentID;
    string stixID;
}
```

Figure 28: Struct type

```
//Store/Retrieve STIX ID through user interaction
//key = attribute, value = AttackPatternID struct (stixID & attribute)
mapping (string => AttackPatternID) attack_pattern_list;

//Retrieve IPFS address
//key = stixID, value = DataLocation struct (contentID & stixID)
mapping(string=> DataLocation) data_locations;
```

Figure 29: Mapping data structure

6.8 DEVELOPMENT PHASE 6: CREATION OF STIX OBJECT QUERY FUNCTIONALITY

To build upon and utilize the Solidity data stores created within development phase 5 (Section 6.7), both Python and Solidity functions were developed to retrieve standardised CTI from IPFS. During the main menu the second option is to retrieve a STIX object that has been stored to IPFS. For the indicator object, the query value is the indicator object pattern value attribute (See Figure 30), and the rest of the objects require just their name to be entered (Figure 31).

```
1: upload threat intelligence object
2: retrieve threat intelligence object
enter: 2

threat intelligence objects:
1 : indicator
2 : malware
3 : tool
4 : vulnerability
5 : attack pattern
6 : course of action
7 : threat actor
8 : infrastructure
9 : intrusion set
select 1 - 9: 1


enter the file hash, url link or ip address:
```

Figure 30: Retrieve choice (Indicator)

```
1: upload threat intelligence object
2: retrieve threat intelligence object
enter: 2

threat intelligence objects:
1 : indicator
2 : malware
3 : tool
4 : vulnerability
5 : attack pattern
6 : course of action
7 : threat actor
8 : infrastructure
9 : intrusion set
select 1 - 9: 2


enter the malware name:
```

Figure 31: Retrieve choice (Not indicator)

The relevant smart contract that contains the mappings and structs for the object contains 2 functions, one to retrieve the STIX ID from the contract mapping based on the user input, and a second one that retrieves the IPFS content address based on the retrieved STIX ID. The object stored on IPFS is then parsed back into the STIX class object to verify that it is still valid expression and then displayed to the user (See Figure 32 & 33).

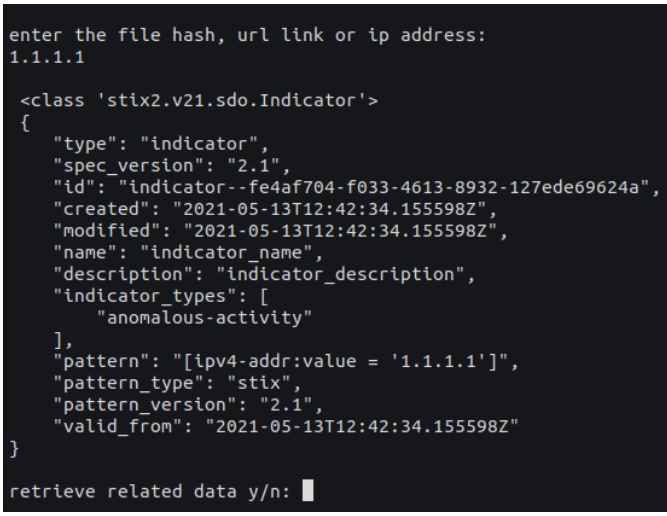


Figure 32: Indicator Object Query

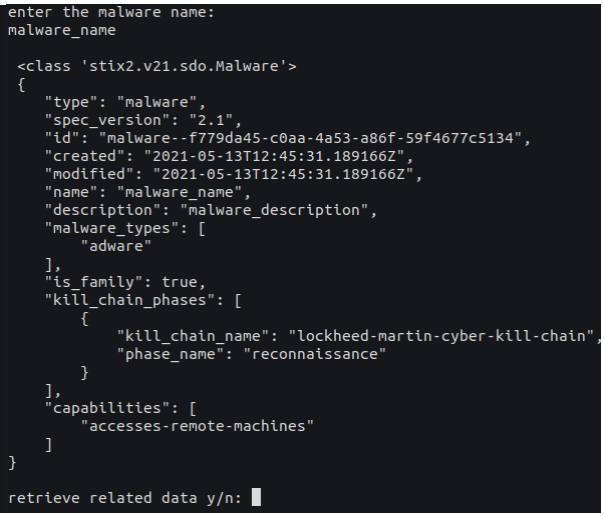


Figure 33: Malware Object Retrieval

6.9 DEVELOPMENT PHASE 7 AND 8: CREATION OF STIX RELATIONSHIP OBJECTS

The relationship STIX object defines a relationship from a target reference to a source reference with a relationship type that indicates the relationship. The relationship object is selected on the upload section of the menu only (See Figure 6 & 30). The application reuses the function to query IPFS for the selected STIX object to return the object ID ready to be stored in a relationship class object. (Figure 34 & 35) The original prototype for adding the relationship required manual input of the relationship types, but as many extensive relationships are depending on the source and target, this was deemed infeasible and would lead to errors in the creation of the STIX relationship object. The RAD methodology showed me quickly that the initial solution was prone to errors due to a lack of user knowledge on each relationship type.

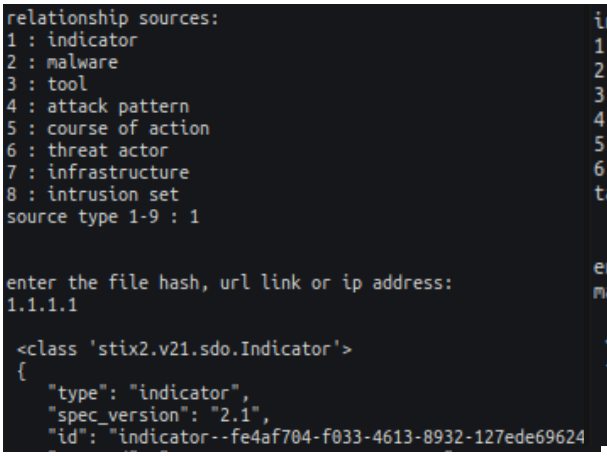


Figure 34: Query relationship source

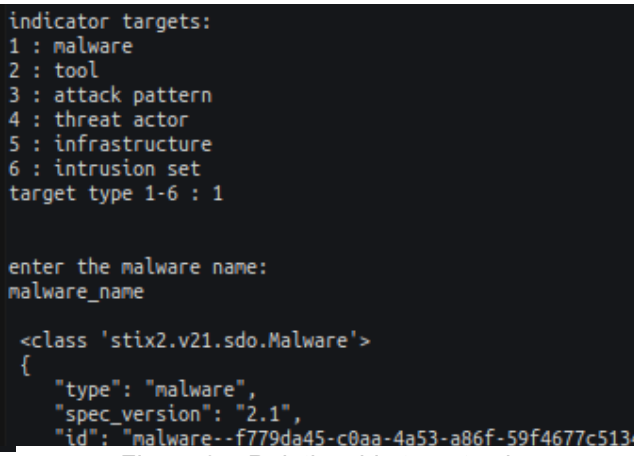


Figure 35: Relationship target value

To set the relationship type based upon the source and target reference values, a complex Python module shows all the objects that a particular source reference can have a relationship with, as can be seen by Figure 35, where there are only 6 compatible objects for an indicator to be the source reference, meaning Threat Actor and Course of Action do not have a relationship directly to an indicator.

Once the source and target values have been selected, this is displayed to the user (See Figure 36), and then the STIX relationship object is created and follows the same upload procedure as seen in development phase 4 and 5 (Figure 37).

Figure 36: Display relationship between objects

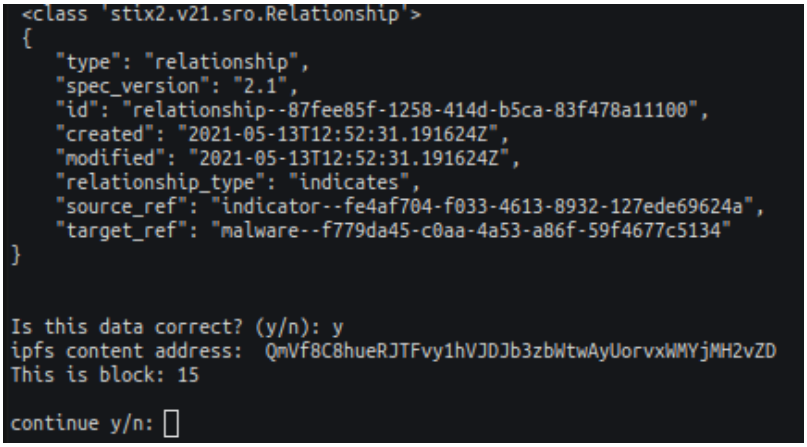


Figure 37 : STIX relationship object uploaded

6.9 DEVELOPMENT PHASE 9: RETRIEVE RELATIONSHIP OBJECTS

The final stage of development for the artefact was to create functionality that will return related relationship data for each given object. This is done after a user queries an object, and an option to retrieve a STIX relationship will appear (See Figure 30 & 31). The STIX ID of the retrieved object is returned and the relationship function is called, passing in the source reference ID. The relationship function communicates with the smart contracts in the same way as development phase 6 (Section 6.8) and will retrieve the associated relationships (See Figure 38). This area of development was the most complex, due to not being able to utilize the same key names multple times within the mapping, meaning that only one relationship object can belong to one source reference.

```
enter the file hash, url link or ip address:
1.1.1.1

 <class 'stix2.v21.sdo.Indicator'>
 {
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--fe4af704-f033-4613-8932-127ede69624a",
    "created": "2021-05-13T12:42:34.155598Z",
    "modified": "2021-05-13T12:42:34.155598Z",
    "name": "indicator_name",
    "description": "indicator_description",
    "indicator_types": [
        "anomalous-activity"
    ],
    "pattern": "[ipv4-addr:value = '1.1.1.1']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T12:42:34.155598Z"
}

retrieve related data y/n: y
indicator--fe4af704-f033-4613-8932-127ede69624a

 <class 'stix2.v21.sro.Relationship'>
 {
    "type": "relationship",
    "spec_version": "2.1",
    "id": "relationship--87fee85f-1258-414d-b5ca-83f478a11100",
    "created": "2021-05-13T12:52:31.191624Z",
    "modified": "2021-05-13T12:52:31.191624Z",
    "relationship_type": "indicates",
    "source_ref": "indicator--fe4af704-f033-4613-8932-127ede69624a",
    "target_ref": "malware--f779da45-c0aa-4a53-a86f-59f4677c5134"
}

continue y/n:
```

Figure 38: Relationship Object Retrieval

7 TESTING

7.1 TESTING METHOD

To evaluate whether the artefact has met the project requirements, the artefact underwent some white box testing techniques, which evaluates the internal structure and the code of a program. White box testing ensures that all independent paths within a module have been run at least once, that all logical decisions are verified and that all loops are executed within their operational bounds.

Aligned with the rapid application development methodology, testing was performed continuously throughout each development phase to ensure functionality and enable continuous improvement of code and development practices. The test cases were derived from the feature specification in section 4.2 to ensure that each feature requirement has been integrated, the logic of the system flows well, and that the code is absent of any typographical or syntactical errors (White Box Testing, 2021) (See Appendix L for test cases and Appendix M for test results).

7.2 EVALUATION OF SOLUTION

7.2.1 RESEARCH

The research for the artefact was critical when developing this application to fully understand the core concept behind completely distributed applications and how they can be used to incorporate data immutability and integrity within a cyber threat intelligence sharing platform, which met the first core objective.

7.2.2 PLANNING, DESIGN AND METHODOLOGY

The use of the rapid-application-development methodology, alongside MoSCoW prioritization, was particularly useful for this project due to it being an unfamiliar project subject with a short time scale on creation. The iterative focus on feature development enabled the use of new project requirements to be developed during the creation of other features while research was ongoing. This was performed through many different rough design prototypes and continuous testing. The static Gantt chart (Appendix A.6) provided a rough guideline for the project timescale, and implemented with the MoSCoW priorities, enabled a focused structure for the artefact.

7.2.3 IMPLEMENTATION

The project met the rest of its core objectives through the implementation, as well as completing all the must and should feature from the feature requirements. Neither the could or wish features were implemented. Legitimate users can enter, store and retrieve cyber threat data that is stored across many different data storage nodes. The project went through many different prototype phases, with the biggest change being in regards to the data storage each time.

It was realised quickly that using a semi-centralized blockchain stood a massive advantage for a consortium of organisations rather than relying on a third-party provider to ensure secure data storage. The decision to create a consortium blockchain rather than a private or public blockchain was realised through the project research and allowed for the system to become semi-centralized, rather than entirely centralized. This comes with both the consortium blockchain and the private IPFS network.

The system utilizes a small sample of the STIX objects that can be uploaded and queried to the distributed store, which was chosen due to the time constraints surrounding the project, as can be seen in feature requirements 2 and 3 (Section 4.2). This limits the range and functionality that comes with the core threat sharing system but enables a very strong basis to build upon. The inclusion of the ability to add relationships between the data became necessary after the beginning of the literature review, in which the clear demarcation between threat information and threat intelligence was defined, being the ability to act upon the contextualized data.

The hardest aspect of the solution was in attempting to utilize the smart contracts to be able to retrieve all relationships that are related to a queried value. This was one of the could priorities, as although it will be an important feature for this project to be actualised into a functional platform, this project aimed to produce a proof of concept of a semi-decentralized and fully distributed cyber threat intelligence platform for permissioned users. Although it has not been implemented, there was research into methods that could enable the implementation of setting and retrieving multiple relationships between each object (See Section 8.2).

8 CONCLUSION

8.1 PROJECT SUMMARY

This project provided an insight into the very current and emerging technologies that come with blockchain applications. Current methods of sharing cyber threat information are messy and unstructured and attempts to subvert that aren't utilised due to a lack of trust with unknown peers. The utilization of the blockchain with distributed storage provides trust in an untrusted environment through immutability. Ensuring data integrity with near-constant availability provides a very strong foundation for a fully functional application to be built upon. As there are only four nodes in this design, if each of the nodes were to be downed, the network would be down completely, which is an important risk to be factored into future works. The data privacy for the project was not entirely secure, and there is little to no confidentiality with using the main IPFS net, but using a private network could lead to complete data loss if all nodes happen to go down.

8.2 FURTHER WORKS

Continuation of this artefact should look into developing either a front-end web design or a local client application for the system to allow legitimate users to interact with the system. After a graphical user interface has been developed, the focus should be on incorporating multiple relationships for each object to produce a report to the user about all related CTI, which would benefit greatly from adding a wider range of CTI sharing attributes to enrichen the threat landscape, allowing users to have much more context for each object.

Some form of data quality measure would be important to verify that the data being input is considered legitimate, which may be implemented in the form of a reputation system. Wu et al proposed a framework called TITAN, which will enhance the trust between peers to help improve the integrity and privacy of a distributed threat sharing system (Wu and Lee, 2019), which is a similar direction in where it needs to go.

to allow for potential scaling, there would need to be a feature in which new users can be added to the system and a method to approve the new user to the system to be able to upload threat intelligence. This would also require fine-grained access control to allow certain channels of communication to be created. Badsha et al. proposed a blockchain-based information sharing model where organizations can achieve fine-grain access control by delegating who may consume their data (Badsha and Vakilinia, 2020).

Total Word Count = 10,255

9 REFERENCES

2016. *Quorum Examples*. https://github.com/ConsenSys/quorum-examples: ConsenSys.

Alien Vault, 2016. *Threat Intelligence Déjà Vu*. Black Hat 2016. [online] Available at: <https://cybersecurity.att.com/blogs/security-essentials/blackhat-2016-threat-intelligence-deja-vu> [Accessed 1 May 2021].

Amoroso, E., 2011. *Cyber attacks*. Burlington, MA: Butterworth-Heinemann.

Amoroso, E., 2011. *Cyber attacks*. Burlington, MA: Butterworth-Heinemann.

Ayoade, G., Karande, V., Khan, L. and Hamlen, K., 2018. Decentralized IoT Data Management Using BlockChain and Trusted Execution Environment. *2018 IEEE International Conference on Information Reuse and Integration (IRI)*,.

Badsha, S. and Vakilinia, I., 2020. BloCyNfo-Share: blockchain-based Cybersecurity Information Sharing with Fine Grained Access Control. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*,.

Barnum, S., 2014. *Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression*. [online] The MITRE Corporation. Available at: <http://www.standardscoordination.org/sites/default/files/docs/STIX_Whitepaper_v1.1.pdf> [Accessed 1 May 2021].

Brown, S., Gommers, J. and Serrano, O., 2015. From Cyber Security Information Sharing to Threat Management. *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*,.

Buterin, V., 2015. *Ethereum Whitepaper*. [online] Available at: <https://ethereum.org/en/whitepaper/> [Accessed 5 May 2021].

Carbon Black, 2020. *2020 Cybersecurity Outlook Report*. [online] vmware. Available at: <2020 Chttps://www.carbonblack.com/resources/2020-cybersecurity-outlook-report/ybersecurity Outlook Report> [Accessed 1 May 2021].

Casino, F. and Patsakis, C., 2019. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, [online] 36, pp.55-81. Available at: <https://www.sciencedirect.com/science/article/pii/S0736585318306324> [Accessed 7 May 2021].

Cichonski, P., 2012. Recommendations of the National Institute of Standards and Technology. *Computer Security Incident Handling Guide*, [online] (800-61). Available at: <http://dx.doi.org/10.6028/NIST.SP.800-61r2> [Accessed 1 May 2021].

CloudFlare. n.d. *Cloud definition*. [online] Available at: <https://www.cloudflare.com/en-gb/learning/cloud/what-is-the-cloud/> [Accessed 6 May 2021].

Cook, T. and Lee, J., n.d. *DappGuard : Active Monitoring and Defense for Solidity Smart Contracts*. [online] Available at: <http://courses.csail.mit.edu/6.857/201:7/project/23.pdf> [Accessed 5 May 2021].

Csrc.nist.gov. n.d. *Glossary | CSRC*. [online] Available at: <https://csrc.nist.gov/glossary/> [Accessed 3 May 2021].

Cyber Threat Alliance. n.d. *Cyber Threat Alliance (CTA) - Home*. [online] Available at: <https://www.cyberthreatalliance.org/> [Accessed 3 May 2021].

Delos Santos, J., 2020a. *Tools for Waterfall Project Management*. [online] Project Management. Available at: <https://project-management.com/3-best-tools-for-waterfall-project-management/> [Accessed 5 May 2021].

Delos Santos, J., 2020b. *Agile Software & Tools for Project Management*. [online] Project Management. Available at: <https://project-management.com/agile-software-tools/> [Accessed 5 May 2021].

Docplayer.net. 2016. *CTI and lessons from law enforcement*. [online] Available at: <https://docplayer.net/2495633-Cyber-threat-intelligence-and-the-lessons-from-law-enforcement-kpmg-com-cybersecurity.html> [Accessed 2 May 2021].

Dyatko, N., 2019. No, you don't store data on the blockchain – here's why. [Blog] Available at: <https://jaxenter.com/blockchain-data-164727.html> [Accessed 1 May 2021].

Eth-brownie. n.d. *Brownie 1.14.6 documentation*. [online] Available at: <https://eth-brownie.readthedocs.io/en/stable/> [Accessed 6 May 2021].

exchange.xforce.ibmcloud.com. n.d. *IBM X-Force Exchange*. [online] Available at: <https://exchange.xforce.ibmcloud.com/> [Accessed 1 May 2021].

Gong, S. and Lee, C., 2020. BLOCIS: Blockchain-Based Cyber Threat Intelligence Sharing Framework for Sybil-Resistance. *Electronics*, 9(3), p.521.

GoQuorum, C., n.d. *Consortium - GoQuorum*. [online] docs.goquorum.consensys.net. Available at: <https://docs.goquorum.consensys.net/en/stable/Concepts/Security/Framework/GoQuorumNetworkSecurity/Consortium/> [Accessed 6 May 2021].

Hasan, R. and Anwar, Z., 2005. A survey of peer-to-peer storage techniques for distributed file systems. *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*,.

Homan, D., Shiel, I. and Thorpe, C., 2019. A New Network Model for Cyber Threat Intelligence Sharing using Blockchain Technology. *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*,.

Hoy, M., 2017. An Introduction to the Blockchain and Its Implications for Libraries and Medicine. *Medical Reference Services Quarterly*, 36(3), pp.273-279.

IPFS Docs. n.d. *IPFS Documentation*. [online] Available at: <https://docs.ipfs.io/> [Accessed 1 May 2021].

Joint Chiefs of Staff, 2017. *JP 2-01, Joint and National Intelligence Support to Military Operations*.

Kampanakis, P., 2014. Security Automation and Threat Information-Sharing Options. *IEEE Security & Privacy*, 12(5), pp.42-51.

Kissflow. 2021. *Rapid Application Development*. [online] Available at: <https://kissflow.com/low-code/rad/rapid-application-development/> [Accessed 6 May 2021].

Li, Z., Wang, W. and Wu, Q., 2020. Blockchain-Based Dynamic Spectrum Sharing for 5G and Beyond Wireless Communications. *Communications in Computer and Information Science*, pp.575-587.

Liew, A., 2007. Understanding Data, Information, Knowledge And Their Inter-Relationships. *Journal of Knowledge Management Practice*, [online] 8(2). Available at: <http://www.tlainc.com/articl134.htm> [Accessed 3 May 2021].

makerdao, 2020. Ethereum 2.0 Gas Issues and Scale. Available at: <https://blog.makerdao.com/how-ethereum-2-0-will-address-gas-issues-and-enable-dai-and-defi-to-scale/> [Accessed 8 May 2021].

Mandiant Cyber Threat Intelligence | FireEye. n.d. *FireEye*. [online] Available at: <https://www.fireeye.com/mandiant/threat-intelligence.html> [Accessed 1 May 2021].

Massessi, D., 2018. *Public Vs Private Blockchain In A Nutshell*. [online] Medium. Available at: <https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f> [Accessed 5 May 2021].

Medium. 2016. *Introduction to Events and Logs in Ethereum*. [online] Available at: <https://media.consensys.net/technical-introduction-to-events-and-logs-in-ethereum-a074d65dd61e> [Accessed 8 May 2021].

Medium. 2018. De*centralized storage transfer speeds faster?*. [online] Available at: <https://medium.com/@ppio/why-is-decentralized-storage-transfer-speeds-faster-c147060f4f0b> [Accessed 2 May 2021].

Medium. 2020. *Hybrid And Federated Blockchain Networks*. [online] Available at: <https://medium.com/xord-one/hybrid-and-federated-blockchain-networks-4508624f10c4#:~:text=In%20a%20Federated%20Blockchain%20network%2C%20multiple%20entities%20make%20use%20of,the%20network%20for%20their%20benefit> [Accessed 4 May 2021].

Mitre, 2012. *Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIXTM)*. [online] Available at: <https://www.mitre.org›sites›files›publications> [Accessed 1 May 2021].

n.d. *Databases on Azure*. https://azure.microsoft.com/en-us/product-categories/databases/: Microsoft.

n.d. *Hyperledger Fabric*. https://hyperledger-fabric.readthedocs.io/en/release-2.2/blockchain.html: Hyperledger.

n.d. *IOC-Finder*. https://www.fireeye.com/services/freeware/ioc-finder.html: FireEye.

n.d. *MongoDB Atlas*. https://www.mongodb.com/cloud/atlas: MongoDB.

n.d. *Remix IDE*. https://remix-project.org/: Remix.

n.d. *Swarm Distributed Storage*. https://ethersphere.github.io/swarm-home/: Ethereum Foundation.

n.d. *web3js*. https://web3js.readthedocs.io/en/v1.3.4/ : Ethereum Foundation.

n.d. *web3py*. https://web3py.readthedocs.io/en/stable/: Ethereum Foundation.

n.d. *ZeroNet*. https://zeronet.io/: ZeroNet.

Nakamoto, S., 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*. [online] Available at: <https://bitcoin.org/bitcoin.pdf> [Accessed 2 May 2021].

NIST Special Publication, C., 2016. Guide to Cyber Threat Information Sharing. *NIST Special Publication*, [online] (800-150). Available at: <http://dx.doi.org/10.6028/NIST.SP.800-150> [Accessed 1 May 2021].

Oasis-Open. n.d. *Cyber Threat Intelligence Technical Committee*. [online] Available at: <https://oasis-open.github.io/cti-documentation/> [Accessed 2 May 2021].

Palmer, S., n.d. Docker vs Vagrant. [Blog] *DevTeam.Space*, Available at: <https://www.devteam.space/blog/docker-vs-vagrant-which-is-better-for-development/> [Accessed 10 May 2021].

Pouwelse, J., Garbacki, P. and Epema, D., 2005. The Bittorrent P2P File-Sharing System: Measurements and Analysis. *Peer-to-Peer Systems IV*, pp.205-216.

Productplan.com. n.d. *MoSCoW Prioritization*. [online] Available at: <https://www.productplan.com/glossary/moscow-prioritization/> [Accessed 11 May 2021].

Purohit, S. and Calyam, P., 2020. DefenseChain: Consortium Blockchain for Cyber Threat Intelligence Sharing and Defense. *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, [online] Available at: <https://ieeexplore.ieee.org/abstract/document/9223313> [Accessed 8 May 2021].

Raft.github.io. n.d. *Raft Consensus Algorithm*. [online] Available at: <https://raft.github.io/> [Accessed 9 May 2021].

Rayborn, J., n.d. *SECURITY CONTENT AUTOMATION PROTOCOL: IS IT BENEFICIAL?*. [online] East Carolina University. Available at: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj02Ka1t8jwAhWlA2MBHTweAZUQFjAAegQIBRAD&url=http%3A%2F%2Fwww.infosecwriters.com%2FPapers%2FJRayborn_SCAP.pdf&usg=AOvVaw1boocdlBEQDvn-doZM6Y3W> [Accessed 2 May 2021].

Sahrom Abu, M., Rahayu Selamat, S., Ariffin, A. and Yusof, R., 2018. Cyber Threat Intelligence – Issue and Challenges. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(1), p.371.

Schoeman, A., 2015. *Demystifying Threat Intelligence*. [online] Infosecurity Magazine. Available at: <https://www.infosecurity-magazine.com/opinions/demystifying-threat-intelligence/> [Accessed 3 May 2021].

Scrum.org. n.d.*What is Scrum?*. [online] Available at: <https://www.scrum.org/> [Accessed 6 May 2021].

Security Alliance, 2016. The Immutability of Cryptocurrency: Part 1 - Concerning Immutability. Available at: <https://www.secalliance.com/blog/the-immutability-of-cryptocurrency-part-1-concerning-immutability> [Accessed 1 May 2021].

Shkoor, M., 2019. Everything You Need to Know About Public, Private, and Consortium Blockchain. [Blog] *Medium*, Available at: <https://medium.com/swlh/everything-you-need-to-know-about-public-private-and-consortium-blockchain-54821c159c7a> [Accessed 8 May 2021].

SoftwareTestingHelp. 2021. *White Box Testing*. [online] Available at: <https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/> [Accessed 12 May 2021].

Storj Lab. 2018. *Storj Whitepaper*. [online] Available at: <https://github.com/storj/whitepaper> [Accessed 5 May 2021].

Szabo, N., 1997. Formalizing and Securing Relationships on Public Networks. *First Monday*, 2(9).

Tian, F., 2016. An agri-food supply chain traceability system for China based on RFID & blockchain technology. *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*,.

Truffle Suite. n.d. *Truffle Suite*. [online] Available at: <https://www.trufflesuite.com/> [Accessed 6 May 2021].

Us-cert.cisa.gov. 2005. *Cyber Threat Source Descriptions | CISA*. [online] Available at: <https://us-cert.cisa.gov/ics/content/cyber-threat-source-descriptions> [Accessed 3 May 2021].

Warburg, B., 2016. How the blockchain will radically transform the economy. *TED Summit*.

Wharton, G., 2021. *Hiscox Cyber Readiness Report 2021*. [online] Hiscox. Available at: <https://www.hiscox.co.uk/sites/default/files/documents/2021-04/21486-Hiscox-Cyber-Readiness-Report-2021-UK.pdf> [Accessed 1 May 2021].

Wood, G., 2021. *Ethereum: A secure decentralised generalised transaction ledger*. Petersberg Version. [online] Available at: <https://ethereum.github.io/yellowpaper/paper.pdf> [Accessed 12 May 2021].

Wu, Y. and Lee, B., 2019. Towards Improved Trust in Threat Intelligence Sharing using Blockchain and Trusted Computing. *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*,.

Zhang, Y. and Wen, J., 2015. An IoT electric business model based on the protocol of bitcoin. *2015 18th International Conference on Intelligence in Next Generation Networks*,.

Zhang, Y. and Xu, C., 2017. Efficient Public Verification of Data Integrity for Cloud Storage Systems from Indistinguishability Obfuscation. *IEEE Transactions on Information Forensics and Security*, [online] 12(3), pp.676-688. Available at: <https://ieeexplore.ieee.org/document/7752973> [Accessed 8 May 2021].

10 APPENDICIES

APPENDIX A – PROJECT PROPOSAL FORM

Degree Title: FCS	**Student's Name: Frazer Chard**
	Supervisor's Name: Muntadher Sallal
	Project Title/Area: Using blockchain technologies to create a trusted and anonymised permissioned cyber threat intelligence platform.

SECTION 1: PROJECT OVERVIEW

1.1 Problem definition - use <u>one sentence</u> to summarise the problem:

Companies are often reluctant to voluntarily share their cyber threat intelligence with reasons ranging from concerns with revealing too much information to the fear of embarrassment (The Threat Intelligence Hangup: Why Don't Organizations Share?, 2017), which diminishes the effectiveness of information exchange. (Riesco and Larriva-Novo, 2019)

1.2 Project description - briefly explain your project:

I will be creating a permissioned cyber threat intelligence platform based on the blockchain to enable anonymised information exchange. The threat incidents would be managed by aggregating log data from observed threat related log data such as firewalls, IDPS, Antivirus etc.

1.3 Background - provide background information, e.g, client, problem domain:

The clients for this project are a small number of SME security focused businesses who would like to implement a secure information exchange system to compliment the security systems that are already in place; this may include but is not limited to security information event management (SIEM), next generation firewalls(NGFW) and endpoint detection and response(EDR). (eSecurityPlanet, 2020)

1.4 Aims and objectives – what are the aims and objectives of your project?

Review current cyber threat intelligence platforms to develop an understanding of what technologies are already out there.
Create threat intelligence feeds through the aggregation of observed threat related log data from the SMEs involved, and also integrate open source feeds.
Identify the best practices to standardise the threat intelligence data to ensure consistency between the variety of sources.
Create a permissioned blockchain network to implement the threat intelligence platform
Determine the most effective way to anonymise the data while also ensuring trust and non-repudiation within the system.

SECTION 2: ARTEFACT

2.1 What is the artefact that you intend to produce?

The artefact that I am going to be producing is a proof of concept for the software and architecture that underpins the blockchain threat intelligence network, which will include the threat intelligence feeds that will aggregate the log data. This will also include smart contracts, which are self-executing code on the blockchain which handles the business logic without the need for manual intervention.

2.2 How is your artefact actionable (i.e., routes to exploitation in the technology domain)?

This artefact is actionable as it will enable me to design and create a blockchain network focusing on factors of trust, non-repudiation and immutability. Using the smart contracts will provide a trusted manner for the secure exchange of information between companies and the threat log feeds.

SECTION 3: EVALUATION

3.1 How are you going to evaluate your work?

I will be evaluating my project by testing a few different factors that work towards solving the issue of companies being reluctant to share threat related information. All the data being uploaded or downloaded should be anonymised to ensure that the identity of the users cannot be revealed while also ensuring non-repudiation to make sure that the data can be verified to be accurate as well. I will also ensure that the blockchain is permissioned and will only be accessible to certain people which should also put the users at ease knowing that any threat related data won't be made publicly available to whoever wants to use it, whether that is competition or threat actors.

3.2 Why is this project honours worthy?

The project I will be undergoing is honours worthy as it is a complex enough project to the point where it covers areas that have not been focused on so heavily during the previous units of the course. It is not too complex though, that it becomes unfeasible to complete within the time-frame allocated for the project. It will require a lot of research to ensure that the system is as validated, secure and anonymised as it should be.

3.3 How does this project relate to your degree title outcomes?

This project relates to Forensic Computing & Security as I am defining an issue within cyber security that relates to the sharing of threat log data gathered from a variety of security tools like firewalls, endpoint detection and response and antivirus for example. I will be solving these issues by designing and building a secure solution that will incorporate knowledge learnt during this course.

3.4 How does your project meet the BCS Undergraduate Project Requirements?

My project meets the BCS Undergraduate Project requirements as I will be applying practical and analytical skills I have learnt throughout my course while also using some older technologies and utilizing them in a different way than has commonly been done before. This will be possible by synthesising a lot of research into these technologies and then applying them to the real need of secure and anonymised threat intelligence, which helps companies minimize the surface of attack and provide better defence (Jelen, 2019). This will be self-managed using a Gantt chart to give me a plan and schedule to stick to.

3.5 What are the risks in this project and how are you going to manage them?

There are some risks that may come about when completing this project. Poor time management could be an issue if there is no plan for how long each task should complete as smaller, less significant tasks could potentially take longer if it hasn't been planned adequately. To manage this, I will be following the Gantt chart that I will be creating, which will give me guidance.

Another potential risk is that I'm not massively confident on a wide range of programming languages, which will be used to create my network, smart contracts and the interconnections between all of these. This will be managed by using programming languages I am more confident in like Java or Python and doing a lot of research into the creation of these items. I will also initially create pseudocode which will also help with the base idea and should allow me to create it regardless of the language used.

SECTION 4: REFERENCES

4.1 Please provide references if you have used any.

Bankinfosecurity.asia. 2017. *The Threat Intelligence Hangup: Why Don't Organizations Share?*. [online] Available at: <https://www.bankinfosecurity.asia/blogs/threat-intelligence-hangup-dont-organizations-share-p-2465> [Accessed 6 May 2021].

Riesco, R. and Larriva-Novo, X., 2019. Cybersecurity threat intelligence knowledge exchange based on blockchain. *Telecommunication Systems*, 73(2), pp.259-288.

ESecurityPlanet. 2020. *Top Threat Intelligence Platforms for 2021 | eSecurity Planet*. [online] Available at: <https://www.esecurityplanet.com/products/threat-intelligence-platforms/> [Accessed 5 May 2021].

Jelen, S., 2019. *SecurityTrails | Cyber Threat Intelligence*. [online] Securitytrails.com. Available at: <https://securitytrails.com/blog/cyber-threat-intelligence> [Accessed 6 May 2021].

Deloitte. n.d. *Blockchain Security Risks for Financial Organizations | Deloitte US*. [online] Available at: <https://www2.deloitte.com/us/en/pages/risk/articles/blockchain-security-risks.html> [Accessed 6 May 2021].

SECTION 5: ETHICS

5.1 Have you submitted online ethics checklist to your supervisor? **Yes**

5.2 Has the checklist been approved by your supervisor? **Yes**

SECTION 6: PROPOSED PLAN (GANTT CHART)

Dissertation
S5115137
Project Lead: Frazer Chard
Today's Date: 5/9/2021
Start Date: 01/02/21
First Day of Week (Sun=1): 2

WB	Tasks	Start	End	Duration (Days)	% Complete	Days Complet	Days Remain
1	Investigation	01/02/21	02/05/21	90	100%	90	0
1.1	Research into the cyber threat landsc	01/02/21	22/04/21	80	100%	80	0
1.2	Research threat intelligence sources	02/02/21	04/03/21	30	100%	30	0
1.3	Research threat intelligence sharing s	02/02/21	03/04/21	60	100%	60	0
1.4	Research into blockchain technology	03/02/21	14/04/21	70	100%	70	0
1.5	Research into Etheruem and smart co	03/02/21	04/04/21	60	100%	60	0
1.6	Research into distributed storage	03/03/21	27/04/21	55	100%	55	0
2	Analysis	16/02/21	07/04/21	50	100%	50	0
2.1	Compare and choose best languages	16/02/21	26/02/21	10	100%	10	0
2.2	Decide what smart contracts will be n	16/02/21	28/03/21	40	100%	40	0
2.3	Compare and choose blockchain syst	16/02/21	04/03/21	16	100%	16	0
2.4	Compare and choose consensus mec	19/02/21	28/02/21	9	100%	9	0
2.5	Choose a valid standardising languag	23/02/21	07/03/21	12	100%	12	0
2.6	Decide on the best storage method	23/02/21	20/03/21	25	100%	25	0
3	Design	20/02/21	11/04/21	50	100%	50	0
3.1	Initial System Design	20/02/21	11/03/21	19	100%	19	0
3.2	Secondary System Design	09/03/21	23/03/21	14	100%	14	0
3.3	Final System Design	22/03/21	12/04/21	21	100%	21	0
4	Implementation	08/03/21	12/05/21	65	100%	65	0
4.1	Create blockchain network	09/03/21	19/04/21	41	100%	41	0
4.2	Create smart contracts	20/03/21	04/05/21	45	100%	45	0
4.3	Connect to blockchain and distributed	21/03/21	25/04/21	35	100%	35	0
4.4	Enable upload and retrieval of threat i	24/03/21	13/05/21	50	100%	50	0
5	Testing	09/04/21	24/05/21	45	100%	45	0
5.1	Test upload to data store	09/04/21	16/04/21	7	100%	7	0
5.2	Test data conversion	16/04/21	23/04/21	7	100%	7	0
5.3	Test data upload via blockchain	23/04/21	30/04/21	7	100%	7	0
5.4	Test retrieving related data	30/04/21	07/05/21	7	100%	7	0
6	Evaluation	07/05/21	14/05/21	7	100%	7	0

APPENDIX B - ETHICS CHECKLIST

About Your Checklist	
Ethics ID	34575
Date Created	18/11/2020 20:51:20
Status	Approved
Date Approved	05/02/2021 11:50:02
Date Submitted	18/11/2020 21:15:39
Risk	Low

Researcher Details	
Name	Frazer Chard
Faculty	Faculty of Science & Technology
Status	Undergraduate (BA, BSc)
Course	BSc (Hons) Forensic Computing & Security
Have you received funding to support this research project?	

Project Details	
Title	Using blockchain technologies to create a trusted and anonymised permissioned cyber threat intelligence platform.
Start Date of Project	30/09/2020
End Date of Project	14/05/2021
Proposed Start Date of Data Collection	01/02/2021
Supervisor	Muntadher Sallal
Approver	Muntadher Sallal
Summary - no more than 600 words (including detail on background methodology, sample, outcomes, etc.)	
I will be creating a cyber threat intelligence sharing platform that will allow the participants to share any threat-related data that will be gathered from security tools as well open-source threat-related data feeds. This will be done performed through a distributed ledger that records each upload, in which every user will have a copy of to ensure trust without the need to actually trust the other users.	

Filter Question: Is your study solely literature based?

Additional Details	
Will you have access to personal data that allows you to identify individuals which is not already in the public	No
domain?	
Will you have access to confidential corporate or company data (that is not covered by confidentiality terms within an agreement or separate confidentiality agreement)?	No

Storage, Access and Disposal of Research Data	
Once your project completes, will any anonymised research data be stored on BU's Online Research Data Repository "BORDaR"?	Yes

APPENDIX C - PROJECT FIRST PROGRESS REVIEW

To be completed and signed by the Supervisor and Student during week **commencing 8 March 2021**.

Student: Frazer Chard	**Supervisor: Muntadher Sallal**

Assessment

1. Definition of the problem *Has the problem been defined, has the artefact been identified and have objectives been set?*	Yes
Comments:	
2. Review of literature and related work *Is there evidence of appropriate research?*	To some extent
Comments:	
3. Methodology and Artefact *Is there evidence of appropriate analysis of the problem and design of a solution and appropriate evaluation?*	Yes
Comments: He is achieved a good progress in project implementation.	
4. Dissertation *Have sections of the dissertation been written and has the Supervisor seen these?*	To some extent
Comments:	
5. Planning & Progress *Is there an acceptable plan for this project and is it being followed?*	Yes
Comments:	
6. Proposal & Online Ethics Checklist *Are proposal and ethic checklist submitted? Are they approved?*	Yes, both are submitted and appr
7. Overall Assessment	Satisfactory
Signed: Supervisor: ……*MuntadherSallal*……Student: ……FrazerChard……..... Date: ……09/03/2021……	

- Supervisor to retain the signed form and supply the student with a copy if required.
- Supervisor to upload the form on Brightspace and grade as *Satisfactory*, *Requires Major Improvement, Requires Minor Improvement, Unsatisfactory or Invalid*.
- Supervisor to notify the Project Coordinator if the student is at risk of failing the project or not engaging.

APPENDIX D – STIX 1.0 CORE COMPONENTS

Schema	Version	Schema	Documentation
Core	1.2	XSD	HTML
Common	1.2	XSD	HTML
Default Vocabularies	1.2.0	XSD	HTML
Data Marking	1.2	XSD	HTML
Campaign	1.2	XSD	HTML
Course of Action	1.2	XSD	HTML
Exploit Target	1.2	XSD	HTML
Incident	1.2	XSD	HTML
Indicator	2.2	XSD	HTML
Report	1.0	XSD	HTML
Threat Actor	1.2	XSD	HTML
TTP	1.2	XSD	HTML

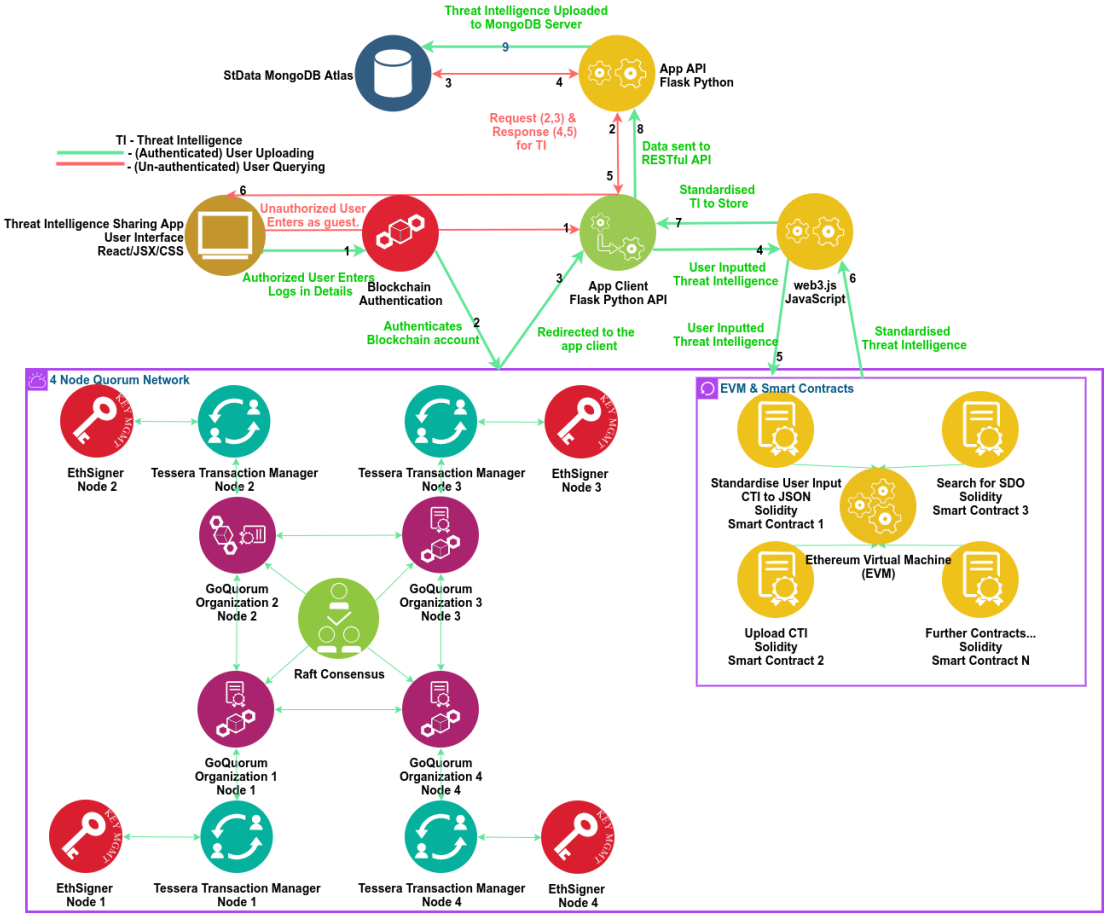
(Barnum, 2014)

APPENDIX E – STIX 1.0 EXTENSIONS

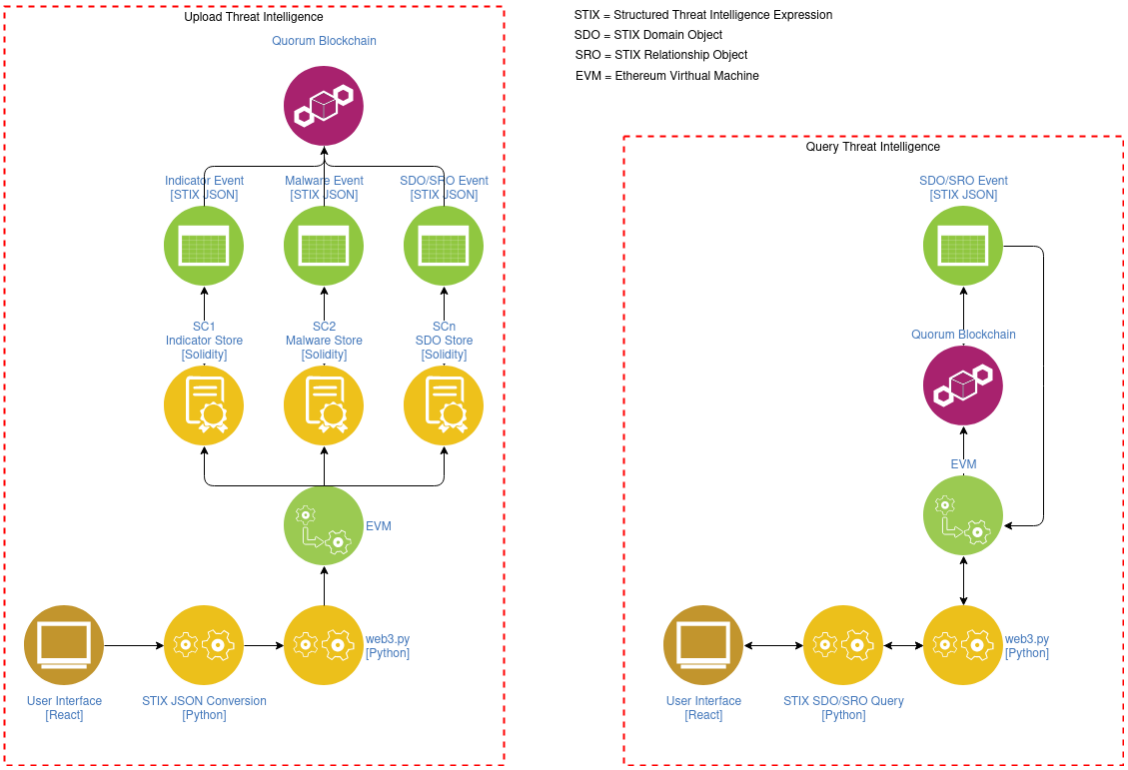
Schema	Extension Point	Version	Schema	Documentation
CIQ 3.0	Address	1.2	XSD	HTML
CAPEC 2.7	Attack Pattern	1.1	XSD	HTML
CIQ 3.0	Identity	1.2	XSD	HTML
MAEC 4.1	Malware	1.1	XSD	HTML
Simple Marking	Marking	1.2	XSD	HTML
Terms of Use	Marking	1.1	XSD	HTML
TLP	Marking	1.2	XSD	HTML
Generic Structured COA	Structured COA	1.2	XSD	HTML
Generic Test Mechanism	Test Mechanism	1.2	XSD	HTML
OpenIOC	Test Mechanism	1.2	XSD	HTML
OVAL 5.10	Test Mechanism	1.2	XSD	HTML
Snort	Test Mechanism	1.2	XSD	HTML
Yara	Test Mechanism	1.2	XSD	HTML
CVRF	Vulnerability	1.2	XSD	HTML

(Barnum, 2014)

APPENDIX F – CLOUD SOLUTION



APPENDIX G – UTILIZING EVENTS FOR STORAGE



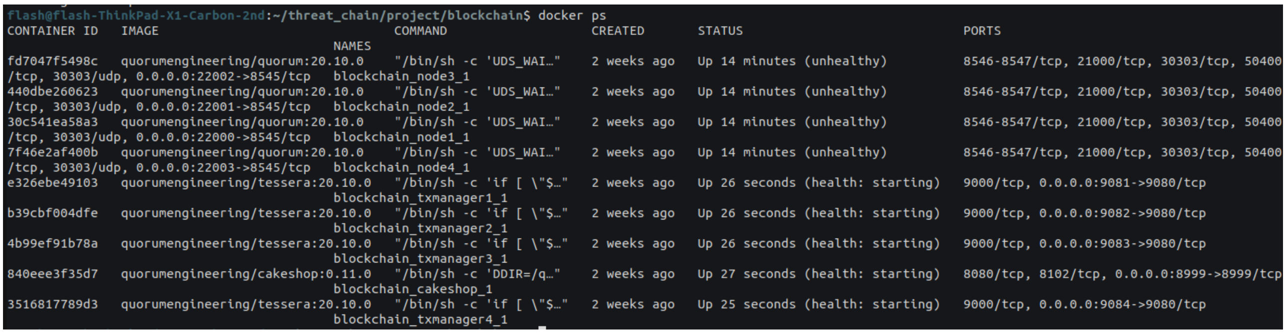
APPENDIX H – QUORUM NETWORK ENVIRONMENT

```
flash@flash-ThinkPad-X1-Carbon-2nd:~/threat_chain/project/blockchain$
QUORUM_CONSENSUS=raft docker-compose up -d
blockchain_txmanager2_1 is up-to-date
blockchain_txmanager3_1 is up-to-date
blockchain_cakeshop_1 is up-to-date
blockchain_txmanager4_1 is up-to-date
blockchain_txmanager1_1 is up-to-date
blockchain_node2_1 is up-to-date
blockchain_node3_1 is up-to-date
blockchain_node4_1 is up-to-date
blockchain_node1_1 is up-to-date
flash@flash-ThinkPad-X1-Carbon-2nd:~/threat_chain/project/blockchain$ docker-compose start
Starting txmanager1 ... done
Starting node1      ... done
Starting txmanager2 ... done
Starting node2      ... done
Starting txmanager3 ... done
Starting node3      ... done
Starting txmanager4 ... done
Starting node4      ... done
Starting cakeshop   ... done
```

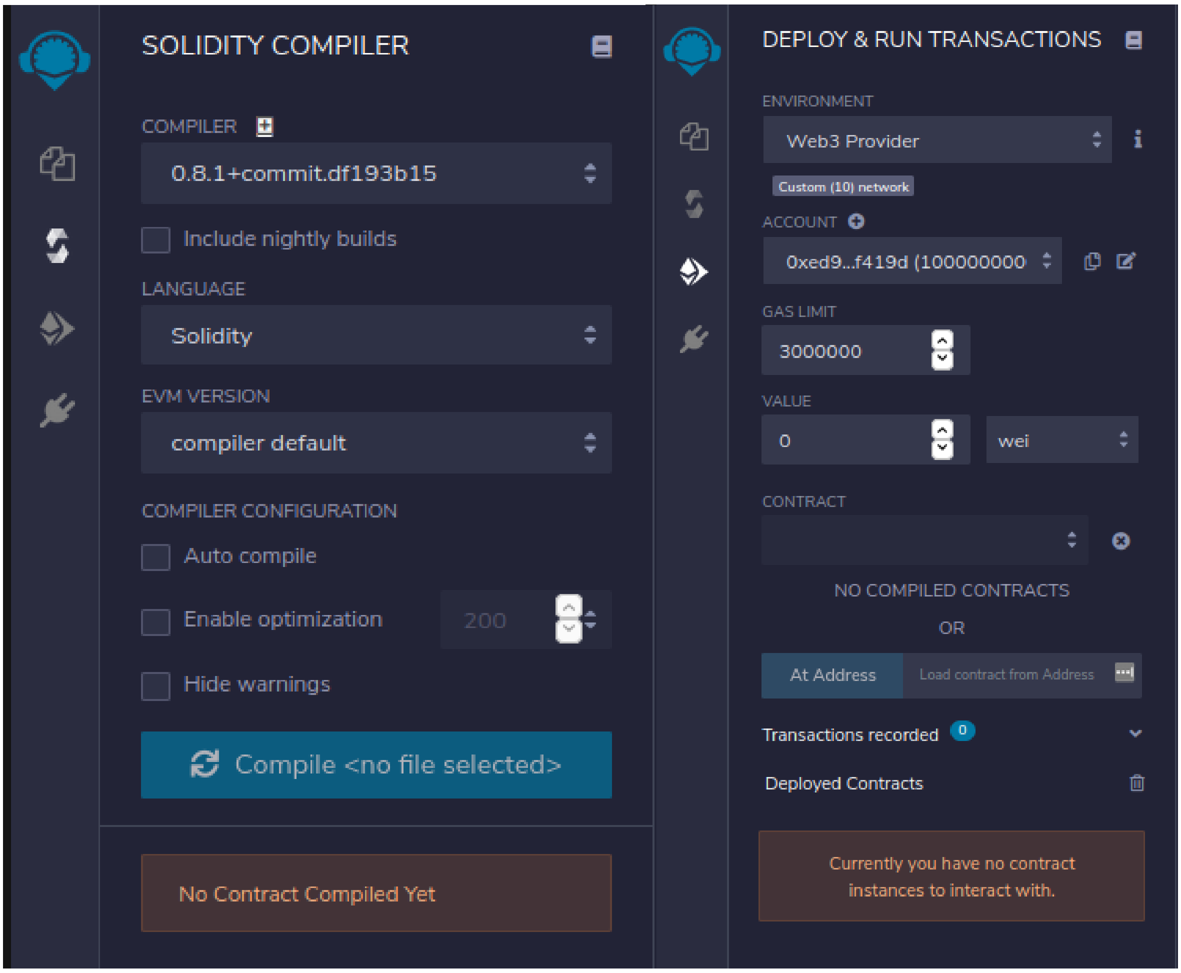
APPENDIX I – IPFS NETWORK ENVIRONMENT

```
flash@flash-ThinkPad-X1-Carbon-2nd:~$ ipfs daemon
Initializing daemon...
go-ipfs version: 0.8.0
Repo version: 11
System version: amd64/linux
Golang version: go1.15.8
2021/05/14 20:09:20 failed to sufficiently increase receive buffer si
ze (was: 208 kiB, wanted: 2048 kiB, got: 416 kiB). See https://github
.com/lucas-clemente/quic-go/wiki/UDP-Receive-Buffer-Size for details.
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/udp/4001/quic
Swarm listening on /ip4/172.16.239.1/tcp/4001
Swarm listening on /ip4/172.16.239.1/udp/4001/quic
Swarm listening on /ip4/172.17.0.1/tcp/4001
Swarm listening on /ip4/172.17.0.1/udp/4001/quic
Swarm listening on /ip4/172.18.0.1/tcp/4001
Swarm listening on /ip4/172.18.0.1/udp/4001/quic
Swarm listening on /ip4/192.168.0.54/tcp/4001
Swarm listening on /ip4/192.168.0.54/udp/4001/quic
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /ip6/::1/udp/4001/quic
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/udp/4001/quic
Swarm announcing /ip4/192.168.0.54/tcp/4001
Swarm announcing /ip4/192.168.0.54/udp/4001/quic
Swarm announcing /ip4/86.27.205.91/tcp/42376
Swarm announcing /ip4/86.27.205.91/udp/42376/quic
Swarm announcing /ip6/::1/tcp/4001
Swarm announcing /ip6/::1/udp/4001/quic
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

APPENDIX J - DOCKER ENVIRONMENT RUNNING



APPENDIX K – REMIX COMPILER, DEPLOYMENT USING WEB3 NODE



APPENDIX L – TEST CASES

TEST #	REQ ID	MoSCoW	TEST DESCRIPTION	EXPECTED OUTCOME	ACTUAL OUTCOME	PASS / FAIL
1	**1**	Must	Attempt to log into the application using correct credentials	Load main menu and displays account key associated to the username	Load main menu and displays account key associated to the username	PASS
2	**1**	Must	Attempt to log into the application using incorrect password and/or username	System will request user to enter the username and password again	System will keep presenting the username and password input forms	PASS
3	**2**	Must	Fill in Indicator upload form	Indicator STIX Object will be created and displayed	Indicator STIX Object will be created and displayed	PASS
4	**2**	Must	Fill in Tool upload form	Tool STIX Object will be created and displayed	Tool STIX Object will be created and displayed	PASS
5	**2**	Must	Fill in Attack Pattern upload form	Attack Pattern STIX Object will be created and displayed	Attack Pattern STIX Object will be created and displayed	PASS
6	**2**	Must	Fill in Malware upload form	Malware STIX Object will be created and displayed	Malware STIX Object will be created and displayed	PASS
7	**3**	Should	Fill in Vulnerability upload form	Vulnerability STIX Object will be created and displayed	Vulnerability STIX Object will be created and displayed	PASS
8	**3**	Should	Fill in Course of Action upload form	Course of Action STIX Object will be created and displayed	Course of Action STIX Object will be created and displayed	PASS
9	**3**	Should	Fill in Threat Actor upload form	Threat Actor STIX Object will be created and displayed	Threat Actor STIX Object will be created and displayed	PASS
10	**3**	Should	Fill in Intrustion Set upload form	Intrusion Set STIX Object will be created and displayed	Intrusion Set STIX Object will be created and displayed	PASS

11	**3**	Should	Fill in Infrastructure upload form	Infrastructure STIX Object will be created and displayed	Infrastructure STIX Object will be created and displayed	PASS
12	**4**	Could	Fill in relevant upload form for the object	STIX object will be created for the selected STIX object	Requirement ID 4 was not met.	FAIL
13	**5**	Must	Upload manual data into STIX class	Will present the object type for the user before printing the object uploading to data store	Presents object type before printing the object for the user	PASS
14	**6**	Must	Upload manual data into STIX class	A content address will be returned from the JSON write function	A content address was returned from the JSON write function	PASS
15	**7**	Must	Confirm the uploaded STIX object is correct	Block number is returned, and user is asked to continue. STIX ID, Content ID and Queriable Attribute uploaded to Smart Contract	Block number is returned, and user is asked to continue. STIX ID, Content ID and Queriable Attribute uploaded to Smart Contract	PASS
16	**8**	Should	Upload relationship between a source object and a target object	STIX relationship object is created	STIX relationship object is created	PASS
17	**9**	Should	Enter a source reference and choose the relationship based on the target	Can select different choices for relationship types dependant on the source and target references	Can select different choices for relationship types dependant on the source and target references	PASS
18	**10** **11**	Must	Retrieve STIX Indicator Object by querying pattern value	STIX Object that has the queried indicator pattern is returned	STIX Object that has the queried indicator pattern is returned	PASS
19	**10** **12**	Should	Retrieve STIX Object (Not Indicator) by querying Object Name	STIX Object that has the queried name attribute is returned	STIX Object that has the queried name attribute is returned	PASS
20	**13**	Could	Select option to return related data after item has been queried	All objects that contain a relationship with the queried object will be returned.	Only one relationship item can be set per object, so only one relationship can be returned.	FAIL

APPENDIX M – TEST CASE RESULTS

TEST #	RESULTS

1A	`enter org name: TCN1`
	`enter password: pass1`
	
	`Welcome TCN1 , your address is 0xed9d02e382b34818e88B88a309c7fe71E65f419d`

1B	`enter org name: TCN2`
	`enter password: pass2`
	
	`Welcome TCN2 , your address is 0xcA843569e3427144cEad5e4d5999a3D0cCF92B8e`

1C	`enter org name: TCN3`
	`enter password: pass3`
	
	`Welcome TCN3 , your address is 0x0fBDc686b912d7722dc86510934589E0AAf3b55A`

1D	`enter org name: TCN4`
	`enter password: pass4`
	
	`Welcome TCN4 , your address is 0x9186eb3d20Cbd1F5f992a950d808C4495153ABd5`

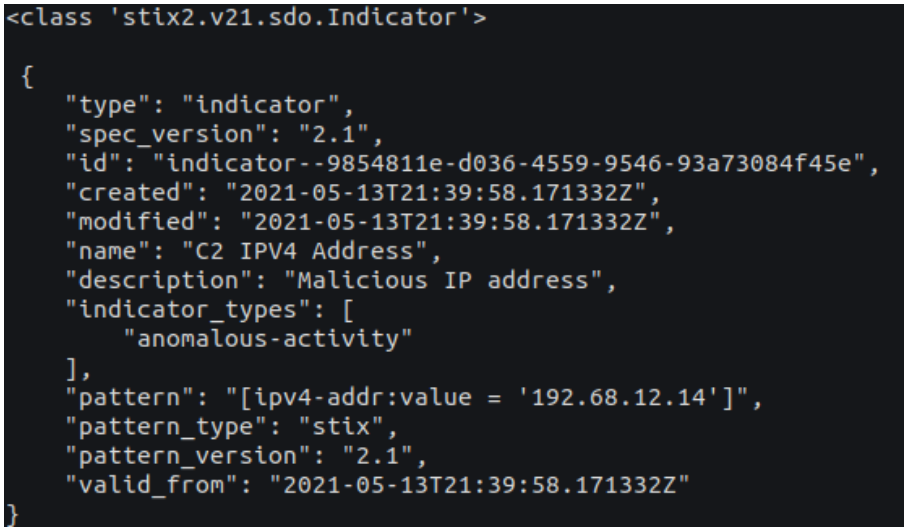
2	`Threat-Chain CLI Application`
	
	`enter org name: bad value`
	`enter password: badpass`
	
	`enter org name: TCN1`
	`enter password: badpass`
	
	`enter org name: bad value`
	`enter password: pass1`
	
	`enter org name:`
	`enter password:`
	
	`enter org name: TCN1`
	`enter password: pass1`

3A	`<class 'stix2.v21.sdo.Indicator'>`
	
	`{`
	`"type": "indicator",`
	`"spec_version": "2.1",`
	`"id": "indicator--422405a1-adbb-4ec9-beb6-a6774611fb91",`
	`"created": "2021-05-13T21:33:57.666401Z",`
	`"modified": "2021-05-13T21:33:57.666401Z",`
	`"name": "C2 URL Domain",`
	`"description": "URL domain for a C2 system",`
	`"indicator_types": [`
	`"attribution"`
	`],`
	`"pattern": "[url:value = 'http://badurl.com']",`
	`"pattern_type": "stix",`
	`"pattern_version": "2.1",`
	`"valid_from": "2021-05-13T21:33:57.666401Z"`
	`}`
	`correct? (y/n): y`

3B

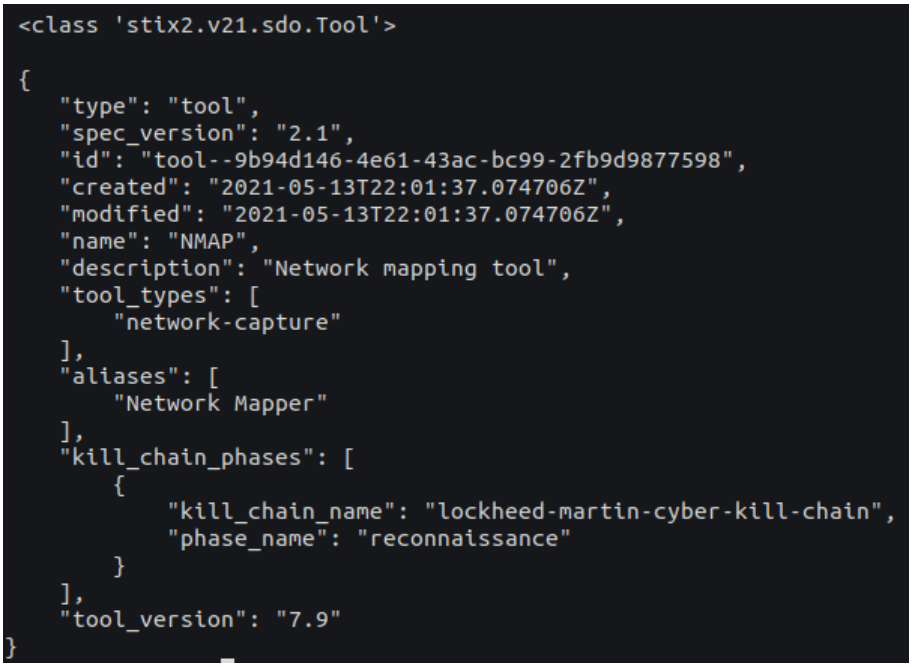
```
<class 'stix2.v21.sdo.Indicator'>

{
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--74f19577-2d1d-4819-8d36-56df55a52361",
    "created": "2021-05-13T21:39:07.915851Z",
    "modified": "2021-05-13T21:39:07.915851Z",
    "name": "Malware File Hash",
    "description": "MD5 File Hash",
    "indicator_types": [
        "malicious-activity"
    ],
    "pattern": "[file:hashes.'md5' = 'd64fde938f4f9c83df63b9da26c0fc26']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T21:39:07.915851Z"
}
```

3C



```
<class 'stix2.v21.sdo.Indicator'>

{
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--9854811e-d036-4559-9546-93a73084f45e",
    "created": "2021-05-13T21:39:58.171332Z",
    "modified": "2021-05-13T21:39:58.171332Z",
    "name": "C2 IPV4 Address",
    "description": "Malicious IP address",
    "indicator_types": [
        "anomalous-activity"
    ],
    "pattern": "[ipv4-addr:value = '192.68.12.14']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T21:39:58.171332Z"
}
```

4



```
<class 'stix2.v21.sdo.Tool'>

{
    "type": "tool",
    "spec_version": "2.1",
    "id": "tool--9b94d146-4e61-43ac-bc99-2fb9d9877598",
    "created": "2021-05-13T22:01:37.074706Z",
    "modified": "2021-05-13T22:01:37.074706Z",
    "name": "NMAP",
    "description": "Network mapping tool",
    "tool_types": [
        "network-capture"
    ],
    "aliases": [
        "Network Mapper"
    ],
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "reconnaissance"
        }
    ],
    "tool_version": "7.9"
}
```

5

<class 'stix2.v21.sdo.AttackPattern'>


```
{
    "type": "attack-pattern",
    "spec_version": "2.1",
    "id": "attack-pattern--9b58fc4e-ecae-41bc-a595-78020057c13e",
    "created": "2021-05-13T22:11:45.457222Z",
    "modified": "2021-05-13T22:11:45.457222Z",
    "name": "Spear Phishing Practiced by X",
    "description": "A method of SP used by group X",
    "aliases": [
        "X SP"
    ],
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "exploitation"
        }
    ],
    "external_references": [
        {
            "source_name": "CAPEC",
            "description": "capec desc",
            "url": "('http://www.capec.com/capec-123',)",
            "external_id": "capec-123"
        }
    ]
}
```


6

<class 'stix2.v21.sdo.Malware'>


```
{
    "type": "malware",
    "spec_version": "2.1",
    "id": "malware--f6297cd1-60c3-4a6c-92a6-b83bad69197f",
    "created": "2021-05-13T22:13:51.274052Z",
    "modified": "2021-05-13T22:13:51.274052Z",
    "name": "Flash Ransomware",
    "description": "IOT ransomware",
    "malware_types": [
        "ransomware"
    ],
    "is_family": false,
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "exploitation"
        }
    ],
    "capabilities": [
        "compromises-data-availability",
        "compromises-data-integrity"
    ]
}
```

7

```
<class 'stix2.v21.sdo.Vulnerability'>

{
    "type": "vulnerability",
    "spec_version": "2.1",
    "id": "vulnerability--d0fce50c-c620-4978-b1d0-5e83a0ab4e63",
    "created": "2021-05-13T22:21:04.006249Z",
    "modified": "2021-05-13T22:21:04.006249Z",
    "name": "CVE-123-456",
    "description": "bad vuln",
    "external_references": [
        {
            "source_name": "cve",
            "url": "('http://cve.com/cve-123-456',)",
            "external_id": "cve-123-456",
            "description": "bad vuln"
        }
    ]
}
```

8

```
<class 'stix2.v21.sdo.CourseOfAction'>

{
    "type": "course-of-action",
    "spec_version": "2.1",
    "id": "course-of-action--4e3dfa76-f2cb-4ae8-b253-ce5636ee3b65",
    "created": "2021-05-13T22:22:46.045101Z",
    "modified": "2021-05-13T22:22:46.045101Z",
    "name": "Secure Ports",
    "description": "Close ports xxxx & xx"
}
```

9

```
<class 'stix2.v21.sdo.ThreatActor'>

{
    "type": "threat-actor",
    "spec_version": "2.1",
    "id": "threat-actor--7afb2fcd-47df-43ea-9fa8-1b29c7e63ea5",
    "created": "2021-05-13T22:39:25.717479Z",
    "modified": "2021-05-13T22:39:25.717479Z",
    "name": "Jack Badguy",
    "description": "Anonymous Hacktivist",
    "threat_actor_types": [
        "activist"
    ],
    "aliases": [
        "JBGUY"
    ],
    "roles": [
        "agent"
    ],
    "goals": [
        "anonymous goals"
    ],
    "sophistication": "advanced",
    "resource_level": "team",
    "primary_motivation": "ideology"
}
```

10


```
<class 'stix2.v21.sdo.IntrusionSet'>

 {
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c8f9c",
    "created": "2021-05-13T22:42:15.822245Z",
    "modified": "2021-05-13T22:42:15.822245Z",
    "name": "IS1",
    "description": "Intrusion set 1",
    "aliases": [
        "IS 12"
    ],
    "goals": [
        "Gain access"
    ],
    "resource_level": "club",
    "primary_motivation": "dominance"
}
```


11


```
<class 'stix2.v21.sdo.Infrastructure'>

 {
    "type": "infrastructure",
    "spec_version": "2.1",
    "id": "infrastructure--0726190c-84ee-4758-b786-8f49a91aab45",
    "created": "2021-05-13T22:44:24.951913Z",
    "modified": "2021-05-13T22:44:24.951913Z",
    "name": "JBD Servers",
    "description": "attackers servers",
    "infrastructure_types": [
        "command-and-control"
    ],
    "aliases": [
        "JBADGUYS Servers"
    ],
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "command and control (C2)"
        }
    ]
}
```

13
14

```
<class 'stix2.v21.sdo.CourseOfAction'>

 {
    "type": "course-of-action",
    "spec_version": "2.1",
    "id": "course-of-action--4e3dfa76-f2cb-4ae8-b253-ce5636ee3b65",
    "created": "2021-05-13T22:22:46.045101Z",
    "modified": "2021-05-13T22:22:46.045101Z",
    "name": "Secure Ports",
    "description": "Close ports xxxx & xx"
}
correct? (y/n): y
ipfs content address:  QmR6wxM31MigmhFY53niweSMR3YxdygLg9ZRBM8mvA8dn8
```

15

```
<class 'stix2.v21.sdo.IntrusionSet'>

 {
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c8f9c",
    "created": "2021-05-13T22:42:15.822245Z",
    "modified": "2021-05-13T22:42:15.822245Z",
    "name": "IS1",
    "description": "Intrusion set 1",
    "aliases": [
        "IS 12"
    ],
    "goals": [
        "Gain access"
    ],
    "resource_level": "club",
    "primary_motivation": "dominance"
}
correct? (y/n): y
ipfs content address:  QmS8pcrG7cMwQrBBD87djr1wtAUWFkcDVRvskTWUn1qZ9W
block:  27

continue y/n: y
```

16

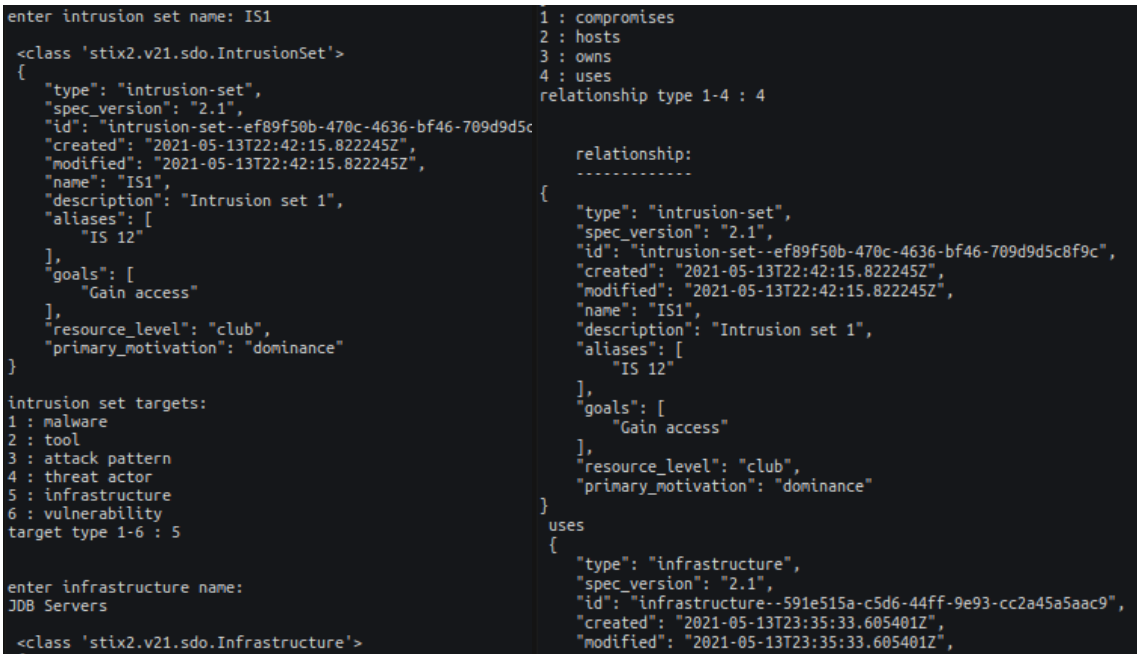
```
<class 'stix2.v21.sro.Relationship'>
{
    "type": "relationship",
    "spec_version": "2.1",
    "id": "relationship--6e36ecfa-71dc-45b7-acbe-bbf10eae58e1",
    "created": "2021-05-13T23:24:01.432765Z",
    "modified": "2021-05-13T23:24:01.432765Z",
    "relationship_type": "uses",
    "source_ref": "threat-actor--7afb2fcd-47df-43ea-9fa8-1b29c7e63ea5",
    "target_ref": "malware--f6297cd1-60c3-4a6c-92a6-b83bad69197f"
}
```

17A

```
relationship sources:
1 : indicator
2 : malware
3 : tool
4 : attack pattern
5 : course of action
6 : threat actor
7 : infrastructure
8 : intrusion set
source type 1-9 : 6


enter threat actor name: Jack Badguy

n <class 'stix2.v21.sdo.ThreatActor'>
{
    "type": "threat-actor",
    "spec_version": "2.1",
    "id": "threat-actor--7afb2fcd-47df-43ea-9fa8-1b29c7e63e
    "created": "2021-05-13T22:39:25.717479Z",
    "modified": "2021-05-13T22:39:25.717479Z",
    "name": "Jack Badguy",
    "description": "Anonymous Hacktivist",
    "threat_actor_types": [
        "activist"
    ],
    "aliases": [
        "JBGUY"
    ],
    "roles": [
        "agent"
    ],
    "goals": [
        "anonymous goals"
    ],
    "sophistication": "advanced",
    "resource_level": "team",
    "primary_motivation": "ideology"
}

threat actor targets:
1 : malware
2 : tool
3 : attack pattern
4 : vulnerability
5 : infrastructure
target type 1-5 : 1


enter the malware name:
Flash Ransomware

 <class 'stix2.v21.sdo.Malware'>
{
    "type": "malware",
    "spec_version": "2.1",
    "id": "malware--f6297cd1-60c3-4a6c-92a6-b83bad69197f",
```

```
    relationship:
    ------------
{
    "type": "threat-actor",
    "spec_version": "2.1",
    "id": "threat-actor--7afb2fcd-47df-43ea-9fa8-1b29c7e63ea5",
    "created": "2021-05-13T22:39:25.717479Z",
    "modified": "2021-05-13T22:39:25.717479Z",
    "name": "Jack Badguy",
    "description": "Anonymous Hacktivist",
    "threat_actor_types": [
        "activist"
    ],
    "aliases": [
        "JBGUY"
    ],
    "roles": [
        "agent"
    ],
    "goals": [
        "anonymous goals"
    ],
    "sophistication": "advanced",
    "resource_level": "team",
    "primary_motivation": "ideology"
}
 uses
 {
    "type": "malware",
    "spec_version": "2.1",
    "id": "malware--f6297cd1-60c3-4a6c-92a6-b83bad69197f",
    "created": "2021-05-13T22:13:51.274052Z",
    "modified": "2021-05-13T22:13:51.274052Z",
    "name": "Flash Ransomware",
    "description": "IOT ransomware",
    "malware_types": [
        "ransomware"
    ],
    "is_family": false,
    "kill_chain_phases": [
        {
            "kill_chain_name": "lockheed-martin-cyber-kill-chain",
            "phase_name": "exploitation"
        }
    ],
    "capabilities": [
        "compromises-data-availability",
        "compromises-data-integrity"
    ]
}
```

17B



```
enter intrusion set name: IS1

 <class 'stix2.v21.sdo.IntrusionSet'>
{
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c
    "created": "2021-05-13T22:42:15.822245Z",
    "modified": "2021-05-13T22:42:15.822245Z",
    "name": "IS1",
    "description": "Intrusion set 1",
    "aliases": [
        "IS 12"
    ],
    "goals": [
        "Gain access"
    ],
    "resource_level": "club",
    "primary_motivation": "dominance"
}

intrusion set targets:
1 : malware
2 : tool
3 : attack pattern
4 : threat actor
5 : infrastructure
6 : vulnerability
target type 1-6 : 5


enter infrastructure name:
JDB Servers

 <class 'stix2.v21.sdo.Infrastructure'>
```

```
1 : compromises
2 : hosts
3 : owns
4 : uses
relationship type 1-4 : 4


    relationship:
    ------------
{
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c8f9c",
    "created": "2021-05-13T22:42:15.822245Z",
    "modified": "2021-05-13T22:42:15.822245Z",
    "name": "IS1",
    "description": "Intrusion set 1",
    "aliases": [
        "IS 12"
    ],
    "goals": [
        "Gain access"
    ],
    "resource_level": "club",
    "primary_motivation": "dominance"
}
 uses
 {
    "type": "infrastructure",
    "spec_version": "2.1",
    "id": "infrastructure--591e515a-c5d6-44ff-9e93-cc2a45a5aac9",
    "created": "2021-05-13T23:35:33.605401Z",
    "modified": "2021-05-13T23:35:33.605401Z",
```

18A

```
threat intelligence objects:
1 : indicator
2 : malware
3 : tool
4 : vulnerability
5 : attack pattern
6 : course of action
7 : threat actor
8 : infrastructure
9 : intrusion set
select 1 - 9: 1


enter the file hash, url link or ip address:
1.1.1.1

 <class 'stix2.v21.sdo.Indicator'>
 {
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--fe4af704-f033-4613-8932-127ede69624a",
    "created": "2021-05-13T12:42:34.155598Z",
    "modified": "2021-05-13T12:42:34.155598Z",
    "name": "indicator_name",
    "description": "indicator_description",
    "indicator_types": [
        "anomalous-activity"
    ],
    "pattern": "[ipv4-addr:value = '1.1.1.1']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T12:42:34.155598Z"
}
```

18 B & C

```
enter the file hash, url link or ip address:
http://badurl.com

 <class 'stix2.v21.sdo.Indicator'>
 {
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--422405a1-adbb-4ec9-beb6-a67746
    "created": "2021-05-13T21:33:57.666401Z",
    "modified": "2021-05-13T21:33:57.666401Z",
    "name": "C2 URL Domain",
    "description": "URL domain for a C2 system",
    "indicator_types": [
        "attribution"
    ],
    "pattern": "[url:value = 'http://badurl.com']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T21:33:57.666401Z"
}
```

```
enter the file hash, url link or ip address:
192.68.12.14

 <class 'stix2.v21.sdo.Indicator'>
 {
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator--9854811e-d036-4559-9546-93a73084f45e",
    "created": "2021-05-13T21:39:58.171332Z",
    "modified": "2021-05-13T21:39:58.171332Z",
    "name": "C2 IPV4 Address",
    "description": "Malicious IP address",
    "indicator_types": [
        "anomalous-activity"
    ],
    "pattern": "[ipv4-addr:value = '192.68.12.14']",
    "pattern_type": "stix",
    "pattern_version": "2.1",
    "valid_from": "2021-05-13T21:39:58.171332Z"
}
```

19

```
enter intrusion set name: IS1

 <class 'stix2.v21.sdo.IntrusionSet'>
 {
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c8f9c",
    "created": "2021-05-13T22:42:15.822245Z",
    "modified": "2021-05-13T22:42:15.822245Z",
    "name": "IS1",
    "description": "Intrusion set 1",
    "aliases": [
        "IS 12"
    ],
    "goals": [
        "Gain access"
    ],
    "resource_level": "club",
    "primary_motivation": "dominance"
}
```

20

```
enter intrusion set name: IS1

 <class 'stix2.v21.sdo.IntrusionSet'>
 {
    "type": "intrusion-set",
    "spec_version": "2.1",
    "id": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c8f9c",
    "created": "2021-05-13T22:42:15.822245Z",
    "modified": "2021-05-13T22:42:15.822245Z",
    "name": "IS1",
    "description": "Intrusion set 1",
    "aliases": [
        "IS 12"
    ],
    "goals": [
        "Gain access"
    ],
    "resource_level": "club",
    "primary_motivation": "dominance"
}

retrieve related data y/n: y

 <class 'stix2.v21.sro.Relationship'>
 {
    "type": "relationship",
    "spec_version": "2.1",
    "id": "relationship--5f231526-4610-4d6b-9388-55229d390d92",
    "created": "2021-05-13T23:36:12.693943Z",
    "modified": "2021-05-13T23:36:12.693943Z",
    "relationship_type": "uses",
    "source_ref": "intrusion-set--ef89f50b-470c-4636-bf46-709d9d5c8f9c",
    "target_ref": "infrastructure--591e515a-c5d6-44ff-9e93-cc2a45a5aac9"
}

continue y/n:
```