SECURE KUBERNETES RESOURCES FROM CPU BASED CRYPTOJACKING

by

RUDRESH NAGENDRAPPA BASAVARAJAPPA, DBA Research Scholar

DISSERTATION

Presented to the Swiss School of Business and Management Geneva

In Partial Fulfilment

Of the Requirements

For the Degree

DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

<MONTH OF GRADUATION, YEAR>

SECURE KUBERNETES RESOURCES FROM CPU BASED CRYPTOJACKING

by

RUDRESH NAGENDRAPPA BASAVARAJAPPA, DBA Research Scholar

APPROVED BY [illegible]

____
Dr. Ljiljana Kukec, Ph.D., Chair

[illegible]

Dr. Saša Petar, Ph.D., Committee Member

[illegible]

____
Dr. Dragan Peraković, Ph.D., Mentor and Committee Member

RECEIVED/APPROVED BY:

____
<Associate Dean's Name, Degree>, Associate Dean

Dedication

This thesis is dedicated to my Parents.

For their endless Love, Support, and Encouragement.

Acknowledgements

There have been a lot of people in my life who have come and gone, and a few who have stayed through the high and lows. This dissertation would not have been possible without a lot of great people guiding and supporting me through the years.

First my Family God "**Shri Guru Thipperudra Swamy**" is my biggest strength and source of guidance. My wife, Bhuvana Poojar, and my children Aayush R Poojar & Aadhya R Poojar gave me reason to continue to try to push myself.

To my parents, who gave me freedom to choose my path very early on, let me deal with challenges and consequences, while still always having my back when I needed them.

My guide and DBA mentor, **Dr. Dragan Peraković** who has been a constant source of guidance and inspiration,

To everyone else who have been helping me realize the good and bad parts of life, and that nothing is permanent.

Thank you. This would not have been possible without any of you.

ABSTRACT

SECURE K8S (KUBERNETES) RESOURCES FROM CPU BASED CRYPTOJACKING

RUDRESH NAGENDRAPPA BASAVARAJAPPA, DBA Research Scholar

2023

Dissertation Chair: <Chair’s Name>

Co-Chair: <If applicable. Co-Chair’s Name>

This research is being conducted to understand the behaviour of the cloud application deployed in the public cloud, private cloud and Blockchain based Hyperledger fabric environments. The main idea is to extract the intension of the cloud resource usage by the application in the generous manner and to derive the expected results that is intended to perform.

Due to the growing demand of the Cryptocurrency usage, the environments used to mine the cryptocurrency is drawing a lot of attention. The environment that is in the form of cloud resources, on prem resources pose a lot of security threats from the hacker in utilizing the hardware resources without the application being disturbed. This results in the increased cost cloud resources billed to the customer.

The tools used in the research methodology to detect the cryptojacking comprises of the open source (Intel V-Tune) and not bind to any hardware vendor that cater the needs of the customer. The goal is to analyse the application on the fly using the tool and collect the data from output of the tools. The data is further cleaned and converted to format (CSV) that is consumed by the necessary deciding logic to further analyse.

The source of the data is driven by the open-source tools used for mining the cryptocurrency. The mining can be done on any popular cloud agonistic platforms that has well defined hardware config. After the application is found to be using the excess hardware that is beyond the threshold value, the method of finding the application usage is done by the method described above.

The challenge in K8S environment to bring the application down without interrupting the operation is well compensated by the ISTIO mesh framework. ISTIO helps in diverting the user traffic for the application to the other instance of the application on the fly. The side car container injection to the K8S pods and configuring the destination rules makes the process smooth in bringing down the affected application.

The results of the study were conducted on the industry standard hardware such has Intel and AMD CPU's. The MSR (Model specific register) configuration plays a vital role in identifying the affected application before and after the application starts.

This research helps the academic people to further enhance the study of different mining application and helps the Enterprise sector to reduce the TCO (Total Cost of Ownership) by identifying the unintended usage of hardware resource by the application.

The results obtained from the research shows that for identifying the cryptojacking the tools used shows positive note. The usage of the tools along with the ease of configuration with respect to the customer application gains upper hand. In short context, the open-source tools selection with hardware platform as abstract on which customer has chosen for the application deployment helps to reduce the TCO.

The results of the study were compared to those of Rupesh Raj Karn et al (2021). The findings of this study may be used by existing academic as a reference in further enhancing the methods of identifying the cryptojacking.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:
INTRODUCTION

This section will provide a high-level overview of the subject under study within the current knowledge set. A background of the subject along with key points will be provided, and the importance of the research will be underscored. Furthermore, the motivation and expected outcomes of the research will be outlined in this section, along with the justification of why the research is important, and why it must be performed. Next, the problem statement will be presented, and later used to propose research questions. Finally, this section will address the gap this research will fill in existing literature.

1.1 Research Background and Scope

Today K8S (Kubernetes) is the widely used standard platform for the application package and deployment. The same is followed by the SME, Enterprise, and Start-up Co using the Cloud based offerings. The Kubernetes (will refer as K8S thru this doc), offers very flexible and easy to deploy the enterprise application packages with existing and new application packages.

A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

Micro services-based architecture is followed by the K8S preliminary design. The different services form the Endpoint for the application to interact with each other. The modularity gives the micro services to alter or upgrade the service alone on the fly while without affecting the existing services. With the recent momentum gaining the Crypto currency acceptance by the different countries, the threat also comes from the Crypto miners to mine the cryptocurrencies. The “Cryptojacking” is the term coined by the industry to mention the same. The mining requires lot of digital hardware with enterprise grade level to mine the crypto currency. Cryptojacking is a type of cyberattack in which hackers exploit a device's computing power without the owner's authorization and use it to mine cryptocurrency.

Dmitry Tanana (2020) says Cryptojacking doesn’t harm infected users directly, victims still suffer losses as their hardware wears out and their processing power works towards interests of cybercriminals instead of their own. The cryptojacking attack is even more harmful for businesses – if a cluster of computers is infected, the business would suffer huge financial losses associated with payment for additional consumed electricity as well as replacing processing units which worked under huge load for a long time. Therefore, early detection of cryptojacking malware is extremely important because it will help to minimize the losses for victims. Malware detection is one of the most relevant tasks in the world of information security.

There are two main approaches to solve this task: statistical analysis and dynamical analysis. Statistical analysis tools usually look for specific signatures in files and compare them to the signatures of known malware within the tool’s library. It is characterized by a high degree of reliability upon detection. Dynamical analysis studies the behavior of an already running program, therefore it is also called behavioral analysis. Both those approaches can be applied toward crypt jacking malware (scmagzine, 2022).

1.2 Research Problem

The Hackers have targeted the less secure cloud deployments as the platforms to introduce the malware and get access to the hardware platforms. By this, the malware will start consuming the Hardware resources (CPU, GPU, FPGA Accelerators, RAM etc.) without the notice of the platform's owners. The intension of the miners is to steal the HW resources.

1.3 Purpose of Research and Questions

The purpose of this research is to generate a better understanding of the risks involved in the cloud environment with little security involved while deploying the applications. The Hackers will come in various form at various levels to make use of the hardware resources without affecting the applications flow.

The challenge for the customers will be dissect the original applications vs affected applications and the method used to resolve the problem.

Research Questions:

- How can the customer identify the applications targeted by miners to make use of the HW platforms?
- What best methods/tools can be are used to identify?
- How can be applications be re- launched without affecting the traffic with zero downtime?

Furthermore, the research will be critical to increase the TCO for the enterprise customers and for academic purpose to go for an extra mile to tackle the mining applications of different variants. By conducting this research now, it is expected to see a higher chance of success in identifying the miners.

1.4 Significance of the Study

There are several challenges that an enterprise customer must overcome when they deploy the application on the public cloud or less secured environments. Although the several measures are taken enough to protect the application from hackers, a small loophole is enough for the jackers to penetrate in several various to gain the access for application. Once the application is taken control, it is very difficult to identify the pattern of the miners used for mining.

This study looks to tackle the challenges faced by enterprise customers in identifying the mining process that affects the customer application.

Further, this research will contribute to the business practices of companies in this space by recommending certain methods or by sharing insights that have been formed by observing and inferring from the experiences of other businesses in the same space.

CHAPTER II:
LITERATURE REVIEW

2.1 Blockchain in K8S

Blockchain technology is widely adopted in various areas of governmental policies such as Banking, Insurance, Healthcare domain. The backbone of Cryptocurrency is the Blockchain tech adopted which is proven for de centralized entity in ensuring the genuine transaction between two parties. Gulenko, A et al (2020).

Banach, R (2019) mentions that the years 2016 and 2017 saw a boom in interest in cryptocurrencies and blockchains. The drive to speculation in Bitcoin specifically, caught the public imagination and caused a bubble in Bitcoin’s value in December 2017.

When a new blockchain application (running on a fresh blockchain) is made live, one of the most common problems encountered is a dearth of ‘hashing power’ or its analogue (according to the consensus protocol used).

Blockchain (Investopedia, 2022) is a distributed digital ledger technology storing the peer-to-peer (P2P) transactions conducted by the parties in the network in an immutable way. Blockchain structure consists of a chain of blocks. As an example, in Bitcoin, each block has two parts: block header and transactions.

A block header consists of the following information:

- Hash of the previous block
- Version,
- Timestamp,
- Difficulty target,
- Nonce
- The root of a Merkle tree.

By inclusion of the hash of the previous block, every block is mathematically bound to the previous one. This binding makes it impossible to change data from any block in the chain. On the other hand, the second part of each block includes a set of individually confirmed transactions.

The power consumption needed to mine a cryptocurrency is for more and needs a special computer that has a good hardware specification. Because of the mining, some countries have restricted the mining as it requires a lot of power consumption. As a matter of fact, the hackers or miners will try to utilize the existing hardware resources by injecting the malware in user application or the Operating systems images that are publicly available.

The list of malware injection does not restrict to only user applications, but it may happen at several levels such as:

- Operating systems Images.
- User Browser.
- Java script code auto execution Etc.

The detection of cryptocurrency malware has been performed by generating its signatures in terms of power consumption, network traffic behavior, operating system processes, and patterns in hardware performance counters

Nukala et al (2021) refers that most of the classic cryptocurrencies such as bitcoin, monero, web chain are built on proof-of-work(pow) algorithm called CryptoNight, which is CPU-bound. They make use of memory bound functions for constructing computational puzzles, in order to maximize profit. This total process requires a lot of disc read and write operations.

Tunde-Onadele et al (2019) states that Emerging container techniques speed up deployments of applications and ease the distribution and delivery of software, but securing containers still has a long way to go toward maturity.

Also, according to Pothula, D.R et al (2019) The runtime container is one of the difficult to secure because threats are happening in real-time when applications are running. Most of the traditional security tools are not designed to monitor run time containers in the production environment.

Hardware prefetcher is a technique in which the processors prefetch data based on the past access behavior by the core. The processor (or the CPU), by using hardware prefetcher, stores instructions from the main memory into the L2 cache. However, on multicore processors, the use of aggressive hardware prefetching causes hampering and results in overall degradation of system performance.

MSR (MSR) registers in processor architecture are used to toggle certain CPU features and computer performance monitoring. By manipulating the MSR registers, hardware prefetchers can be disabled.
A miner running with root privileges can disable the prefetcher. This is done to boost the miner execution performance, thereby increasing the speed of the mining process. We have seen Xmrig miners in our threat intelligence systems using MSR to disable the hardware prefetcher.

Xmrig miners use the RandomX algorithm which generates multiple unique programs that are generated by data selected from the dataset generated from the hash of a key block. The code to be run inside the VM is generated randomly and the resultant hash of its outcome is used as proof of work.

As RandomX programs are run in a VM, this operation is generally memory intensive. Hence, the miner disables the hardware prefetcher using the MSR. According to the documentation of Xmrig, disabling the hardware prefetcher increases the speed up to 15%.

According to Liz Rice (2017) Containers bring many advantages: as described in Docker's original tagline, they allow you to "build once, run anywhere." They do this by bundling together an application and all its dependencies and isolating the application from the rest of the machine it's running on. The containerized application has everything it needs, and it is easy to package up as a container image that will run the same on the laptop, or in a server in a data center. The points which state difference between threat and risk is as follows:

- A risk is a potential problem, and the effects of that problem if it were to occur.
- A threat is a path to that risk occurring.
- A mitigation is a countermeasure against a threat—something you can do to prevent the threat or at least reduce the likelihood of its success.

One way to start thinking about the threat model is to consider the actors involved. These might include:

- External attackers attempting to access a deployment from outside
- Internal attackers who have managed to access some part of the deployment
- Malicious internal actors such as developers and administrators who have some level of privilege to access the deployment
- Inadvertent internal actors who may accidentally cause problems
- Application processes that, while not sentient beings intending to compromise your system, might have programmatic access to the system (Istio, 2022).

Following are the best security principles recommended by (Liz Rice, 2017):

- Least privilege

  One can give can different containers different sets of privileges, each minimized to the smallest set of permissions it needs to fulfil its function

- Defense in depth

  Containers give another boundary where security protections can be enforced.

- Reducing the surface attack

  Splitting a monolith into simple microservices can create clean interfaces between them that may, if carefully designed, reduce complexity, and hence limit the attack surface. There is a counterargument that adding a complex orchestration layer to coordinate containers introduces another attack surface.

According to Liz Rice (2017), A vulnerability is a known flaw in a piece of software that an attacker can take advantage of to perform malicious activity. As a rule, you can assume that the more complex a piece of software is, the more likely it is to have flaws, some of which will be exploitable.

When there is a vulnerability in a common piece of software, attackers may be able to take advantage of it wherever it is deployed, so there is an entire research industry devoted to finding and reporting new vulnerabilities in publicly available software, especially operating system packages and language libraries. You have probably heard of some of the most devastating vulnerabilities, like Shellshock, Meltdown, and Heartbleed, which get not just a name but sometimes even a logo. These are the rock stars of the vulnerability world, but they are a tiny fraction of the thousands of issues that get reported every year.

Dealing with software vulnerabilities is an important aspect of risk management. It's very likely that a deployment of any nontrivial software will include some vulnerabilities, and there is a risk that systems will be attacked through them. To manage this risk, you need to be able to identify which vulnerabilities are present and assess their severity, prioritize them, and have processes in place to fix or mitigate these issues.

To know whether the deployment is running containers with vulnerable software, you need to scan all the dependencies within those containers. There are some different approaches you could take to achieve this.

Imagine a tool that can scan each running container on a host (or across a deployment of multiple hosts). In today's cloud native deployments, it's common to see hundreds of instances of containers initiated from the same container image, so a scanner that takes this approach would be very inefficient, looking at the same dependencies hundreds of times. It's far more efficient to scan the container image from which these containers were derived.

According to Liz Rice (2017), Several tools are available for container image scanning such as Trivy, Clair, and Anchore to commercial solutions from companies like Jogo, Palo Alto, and Aqua. Many container image registry solutions, such as Docker Trusted Registry and the CNCF project Harbor, as well as the registries provided by the major public clouds, include scanning as a built-in feature.

Unfortunately, the results you get from different scanners vary considerably.

If it’s possible for a third-party library to have a bug that an attacker can exploit, the same is true for any code—including the applications that your team is writing. Peer review, static analysis, and testing can all help to identify security issues in your code, but there’s a chance that some issues will slip through. Depending on your organization and the value of its data, there may be bad actors in the world for whom it’s worthwhile trying to find these flaws.

According to Sandro Bartolini et al (2008), Instruction-set extension (ISE) has been widely studied as a means to improve the performance of microprocessor devices running cryptographic applications. It consists, essentially, in endowing an existing processor with a set of additional instructions that can be useful for speeding-up specific cryptographic computations. Recently, researchers became aware of the following: ”The efficiency of an implementation algorithm often depends heavily on the details of the target platform, e.g., on the instruction set or the pipeline of a processor. Instruction-set extension can be better understood only after having clear what an instruction-set is. At the higher level, an instruction-set (or instruction set architecture - ISA) can be defined as the pool of instructions made available by a processor to the assembler programmer or to the compiler. In this sense, the ISA defines a significant quote of the programming interface of the processor: the basic operations that the outside world can ask the processor to do. The whole programming interface of a processor is surely wider than the sole ISA and, in brief, it also encompasses the structure and features of the processor registers, the organization of the memory space, as it is perceived by the programmer (e.g., virtual memory, permissions), and the features of the I/O space. All this information is needed to take full advantage from the features exposed by a microprocessor. Anyway, instructions cannot be arbitrarily complex: a tradeoff has to be done in order to have fast circuits behind the implementation, and the RISC approach (simple, modular, efficient instructions, e.g. MIPS, SPARC, PowerPC) versus the CISC approach (many powerful instructions, e.g. Intel x86) has characterized the main microprocessors on the market. For instance, we will briefly outline some of the features of the ISA of Intelr processors and its evolution through specific extensions during the years. The base ISA of the Intel Pentium-4 class processors [32] is almost the same since the old 386-class processor and is named x86 instruction set. It comprises hundreds of instructions, operating on eight 32-bit general-purpose registers. They can be seen also as the sole 16-bit lower part and four of them even allow using their 16-bit lower part as two 8-bit registers (e.g., AH and AL are 8-bit registers that, together, form the AX 16-bit register, which is the lower part of the 32-bit register EAX). Specific instructions for integer arithmetic, bitwise operations, movement among registers and between registers and memory or I/O space can use 8-, 16- or 32-bit registers. Other instructions manage the program control flow at various levels: jumps, calls and loops, up to interrupt service routine management. This 32-bit ISA has been extended to a 64- bit backward compatible ISA (x86-64) in 2004, after that a similar proposal was done in 2002 by AMD. This extension was motivated by the need to access wide memory regions (i.e., beyond 2-4 GByte, according to the available Operating System) easily, and to support an increased word-level parallelism which was needed by a number of high-end applications.

Since some versions of the 486 model, the processor was extended to natively support floating-point operations, without the need of an external coprocessor. In this way, the programmer sees some additional configuration registers and eight 80-bit registers for working on floating-point operands. Specific instructions move the operands to/from the floating-point register set and trigger floating point computations. Specific circuits implement the register file and the operations that are performed upon instruction execution. Another class of extensions have been proposed, in steps, for vector-like operations which are motivated by the need of supporting efficiently a variety of multimedia applications such as 2-D and 3-D graphics, motion video, image processing, speech recognition, audio synthesis, telephony, and video conferencing. Beginning with the Pentium II and Pentium with Intel MMX technology processor families, a number of incremental extensions have been introduced into the IA-32 architecture to permit IA-32 processors to perform single-instruction multiple-data (SIMD) operations. These extensions include the MMX technology, SSE, SSE2, SSE3 and SSE4 extensions. Each of these extensions provides a group of instructions that perform SIMD operations on packed integer and/or packed floating-point data elements contained in specific registers (64-bit MMX or 128-bit XMM registers). MMX extension allows the programmer to see eight 64-bit register which can be used as groups of eight packed bytes, four packed 16-bit words or two packed 32-bit doublewords. In this way, specific instructions can operate on such vector-like operands more efficiently (e.g., parallel saturating addition on all packed elements) and other instructions are provided for loading/storing values from/to MMX registers. Architecturally, specific circuits for parallel elaboration of SIMD operations are added to the processor, while the MMX register file is shared with the floating-point unit. In this way, no additional storage for MMX registers was needed but, as drawback, big attention has to be taken when using in the same time both floating point and MMX instructions. SSE extension introduce a separated set of eight 128-bit registers (XMM) for SIMD operations, which are intended to support floating-point SIMD operations too. With a dedicated register file, the conflicts with other internal resources, as in case of the floating-point register file used by MMX, are reduced. In particular, each XMM register can be seen by the programmer as four 32-bit single-precision floating-point values. The SIMD operations on such values can help in supporting advanced media and communications applications, for which MMX integer/fixed-point SIMD operations are limiting. Move and conversion instructions too are provided for the interaction of XMM registers with memory, MMX and general-purpose registers. SSE2 extension increase flexibility in using XMM registers as additional floating-point packed values and introduce the support for packed integers too. In fact, each XMM register can be used also as two packed 64-bit double precision floating-point values, 16 packed bytes, eight packed 16-bit words, four packed 32-bit doublewords and two packed 64-bit quadwords. Operations on packed integers into SSE registers allow double parallelism than using MMX and avoid conflicts with the floating-point unit. Additional instructions are provided to operate on this variety of operand types. SSE3 extension enhances the previous instruction set with only 13 instructions that accelerate some SSE, SSE2 and floating-point capabilities. For instance, an optimized floating-point to integer conversion instruction is provided, as well as an unaligned 128-bit load instruction for integer operands, and additional SIMD operations. This highlights the importance of easing the interaction between the existing processor data formats (e.g., integers and float values) and hardware modules from one side, and the extended circuitry (e.g., registers) and operands (e.g., float or integer packed values) on the other side, in order to boost programmability and performance. SSE4 extension was recently proposed with the Intel CoreTM processor family and further improve SSE capabilities. Essentially it improves the flexibility of SSE in supporting compiler vectorization and significantly increases the available packed doubleword computations. This brief story of ISA extensions in Intel processors highlights that the study of the interaction between special hardware resources (e.g., registers, circuits) and the set of available instructions is crucial for accelerating specific computations and thus for the final performance of an ISA extension. It is very interesting to highlight here the implications of ISA extensions towards an Operating System (OS). This occurs mainly when an OS, even minimal, is employed to enable multiprogramming through processor virtualization. In fact, in a multiprogramming system, the physical processor is assigned by the Operating System to the different processes (i.e., running programs) that are contemporarily in execution so that each one is able to execute, even if in an intermittent fashion. The OS switches the process-processor association according to the time spent by the running process (e.g., round robin) or when the latter blocks(e.g., wait for an external event). When a process P1 leaves the processor to the next process P2 (i.e., a context switch happens), the complete processor state has to be saved so that it can be restored later when P1 will be able to continue its execution exactly as if it was never interrupted at all. The state of a process comprises, at least, the value of all processor registers, including the state registers (e.g., flags). Therefore, extending the register organization of a processor implies modifications into the OSs in order to properly manage the machine state. If the OS is not updated, upon a context switch, it saves only the original processor state and it neglects the additional extended registers. In this way, when the process state will eventually be restored on the processor, a part of its state can be corrupted, and the elaboration can become erroneous. This might be considered when designing a solution based on ISE for a target processor for which a number of Operating Systems are already present in the market. All of them have to be updated to support certain ISA extensions. On this point, note that the Intel MMX extension could be supported without OS modifications because they relied on the registers already used by the floating-point unit.

According to Hyperledger (hyperledger-fabric.readthedocs.io, 2023), Fabric has a highly modular and configurable architecture, enabling innovation, versatility, and optimization for a broad range of industry use cases including banking, finance, insurance, healthcare, human resources, supply chain and even digital music delivery.

Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rat her than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed.

The Fabric platform is also permissioned, meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted. This means that while the participants may not *fully* trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off of what trust *does* exist between participants, such as a legal agreement or framework for handling disputes.

One of the most important of the platform's differentiators is its support for pluggable consensus protocols that enable the platform to be more effectively customized to fit particular use cases and trust models. For instance, when deployed within a single enterprise, or operated by a trusted authority, fully byzantine fault tolerant consensus might be considered unnecessary and an excessive drag on performance and throughput. In situations such as that, a crash fault tolerant (CFT) consensus protocol might be more than adequate whereas, in a multi-party, decentralized use case, a more traditional byzantine fault tolerant (BFT) consensus protocol might be required.

Fabric can leverage consensus protocols that do not require a native cryptocurrency to incent costly mining or to fuel smart contract execution. Avoidance of a cryptocurrency reduces some significant risk/attack vectors, and absence of cryptographic mining operations means that the platform can be deployed with roughly the same operational cost as any other distributed system.

The combination of these differentiating design features makes Fabric one of the better performing platforms available today both in terms of transaction processing and transaction confirmation latency, and it enables privacy and confidentiality of transactions and the smart contracts (what Fabric calls "chaincode") that implement them. Hyperledger Fabric has been specifically architected to have a modular architecture.

Whether it is pluggable consensus, pluggable identity management protocols such as LDAP or OpenID Connect, key management protocols or cryptographic libraries, the platform has been designed at its core to be configured to meet the diversity of enterprise use case requirements.

At a high level, Fabric is comprised of the following modular components:

- A pluggable ordering service establishes consensus on the order of transactions and then broadcasts blocks to peers.
- A pluggable membership service provider is responsible for associating entities in the network with cryptographic identities.
- An optional peer-to-peer gossip service disseminates the blocks output by ordering service to other peers.
- Smart contracts (“chaincode”) run within a container environment (e.g. Docker) for isolation. They can be written in standard programming languages but do not have direct access to the ledger state.
- The ledger can be configured to support a variety of DBMSs.
- A pluggable endorsement and validation policy enforcement that can be independently configured per application.

There is fair agreement in the industry that there is no “one blockchain to rule them all”. Hyperledger Fabric can be configured in multiple ways to satisfy the diverse solution requirements for multiple industry use cases.

In a permissionless blockchain, virtually anyone can participate, and every participant is anonymous. In such a context, there can be no trust other than that the state of the blockchain, prior to a certain depth, is immutable. In order to mitigate this absence of trust, permissionless blockchains typically employ a “mined” native cryptocurrency or transaction fees to provide economic incentive to offset the extraordinary costs of participating in a form of byzantine fault tolerant consensus based on “proof of work” (PoW).

Permissioned blockchains, on the other hand, operate a blockchain amongst a set of known, identified and often vetted participants operating under a governance model that yields a certain degree of trust. A permissioned blockchain provides a way to secure the interactions among a group of entities that have a common goal but which may not fully trust each other. By relying on the identities of the participants, a permissioned blockchain can use more traditional crash fault tolerant (CFT) or byzantine fault tolerant (BFT) consensus protocols that do not require costly mining.

Additionally, in such a permissioned context, the risk of a participant intentionally introducing malicious code through a smart contract is diminished. First, the participants are known to one another and all actions, whether submitting application transactions, modifying the configuration of the network or deploying a smart contract are recorded on the blockchain following an endorsement policy that was established for the network and relevant transaction type. Rather than being completely anonymous, the guilty party can be easily identified, and the incident handled in accordance with the terms of the governance model.

A smart contract, or what Fabric calls “chaincode”, functions as a trusted distributed application that gains its security/trust from the blockchain and the underlying consensus among the peers. It is the business logic of a blockchain application.

There are three key points that apply to smart contracts, especially when applied to a platform:

- many smart contracts run concurrently in the network,
- they may be deployed dynamically (in many cases by anyone), and
- application code should be treated as untrusted, potentially even malicious.

Most existing smart-contract capable blockchain platforms follow an **order-execute** architecture in which the consensus protocol:

- validates and orders transactions then propagates them to all peer nodes,
- each peer then executes the transactions sequentially.

The order-execute architecture can be found in virtually all existing blockchain systems, ranging from public/permissionless platforms such as Ethereum (with PoW-based consensus) to permissioned platforms such as Tendermint, Chain, and Quorum.

Smart contracts executing in a blockchain that operates with the order-execute architecture must be deterministic; otherwise, consensus might never be reached. To address the non-determinism issue, many platforms require that the smart contracts be written in a non-standard, or domain-specific language (such as Solidity) so that non-deterministic operations can be eliminated. This hinders wide-spread adoption because it requires developers writing smart contracts to learn a new language and may lead to programming errors.

Further, since all transactions are executed sequentially by all nodes, performance and scale is limited. The fact that the smart contract code executes on every node in the system demands that complex measures be taken to protect the overall system from potentially malicious contracts in order to ensure resiliency of the overall system.

Fabric introduces a new architecture for transactions that we call execute-order-validate. It addresses the resiliency, flexibility, scalability, performance and confidentiality challenges faced by the order-execute model by separating the transaction flow into three steps:

- *execute* a transaction and check its correctness, thereby endorsing it,
- *order* transactions via a (pluggable) consensus protocol, and
- *validate* transactions against an application-specific endorsement policy before committing them to the ledger

This design departs radically from the order-execute paradigm in that Fabric executes transactions before reaching final agreement on their order.

In Fabric, an application-specific endorsement policy specifies which peer nodes, or how many of them, need to vouch for the correct execution of a given smart contract. Thus, each transaction need only be executed (endorsed) by the subset of the peer nodes necessary to satisfy the transaction's endorsement policy. This allows for parallel execution increasing overall performance and scale of the system. This first phase also eliminates any non-determinism, as inconsistent results can be filtered out before ordering.

Because we have eliminated non-determinism, Fabric is the first blockchain technology that enables use of standard programming languages.

As we have discussed, in a public, permissionless blockchain network that leverages PoW for its consensus model, transactions are executed on every node. This means that neither can there be confidentiality of the contracts themselves, nor of the transaction data that they process. Every transaction, and the code that implements it, is visible to every node in the network. In this case, we have traded confidentiality of contract and data for byzantine fault tolerant consensus delivered by PoW.

This lack of confidentiality can be problematic for many business/enterprise use cases. For example, in a network of supply-chain partners, some consumers might be given preferred rates as a means of either solidifying a relationship or promoting additional sales. If every participant can see every contract and transaction, it becomes impossible to maintain such business relationships in a completely transparent network — everyone will want the preferred rates!

As a second example, consider the securities industry, where a trader building a position (or disposing of one) would not want her competitors to know of this, or else they will seek to get in on the game, weakening the trader’s gambit.

In order to address the lack of privacy and confidentiality for purposes of delivering on enterprise use case requirements, blockchain platforms have adopted a variety of approaches. All have their trade-offs.

Encrypting data is one approach to providing confidentiality; however, in a permissionless network leveraging PoW for its consensus, the encrypted data is sitting on every node. Given enough time and computational resource, the encryption could be broken. For many enterprises use cases, the risk that their information could become compromised is unacceptable.

Zero knowledge proofs (ZKP) are another area of research being explored to address this problem, the trade-off here being that, presently, computing a ZKP requires considerable time and computational resources. Hence, the trade-off in this case is performance for confidentiality.

In a permissioned context that can leverage alternate forms of consensus, one might explore approaches that restrict the distribution of confidential information exclusively to authorized nodes.

Hyperledger Fabric, being a permissioned platform, enables confidentiality through its channel architecture and private data feature. In channels, participants on a Fabric network establish a sub-network where every member has visibility to a particular set of transactions. Thus, only those nodes that participate in a channel have access to the smart contract (chaincode) and data transacted, preserving the privacy and confidentiality of both. Private data allows collections between members on a channel, allowing much of the same protection as channels without the maintenance overhead of creating and maintaining a separate channel.

The ordering of transactions is delegated to a modular component for consensus that is logically decoupled from the peers that execute transactions and maintain the ledger. Specifically, the ordering service. Since consensus is modular, its implementation can be tailored to the trust assumption of a particular deployment or solution. This modular architecture allows the platform to rely on well-established toolkits for CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) ordering.

Note also that these are not mutually exclusive. A Fabric network can have multiple ordering services supporting different applications or application requirements.

2.2 Malware Problem & Remedy

The problem of detecting the malware in the container runtime is very overlooked and believed that traditional secure methods are enough. But the hackers are innovating the latest way to get entry into the customer platform without getting noticed. One main problem that malware poses is that the functionality of the user application will not be altered by the malware so that any traditional malware detectors fail in detecting the malware. The only way one can notice is the de graded performance by the application at the peak of its usage.

Rupesh Raj Karn et al (2021), clearly states that in public cloud computing services, access to the hardware resources is typically not available to the customer. Instead, the Linux-kernel system calls at the operating system level can be used as a proxy to signal the possibility of threat in a running container. The system call (syscall) is the fundamental interface between an application and the Linux kernel. A syscall is generated every time the application interacts with the Linux-kernel.

Intel V-Tune Profiler is a performance analysis tool for x86 based machines running Linux or Microsoft Windows operating systems. It can help us optimize application performance, system performance, and system configuration for HPC, cloud, IoT, media, storage, and more, the tool is used to analyze the customer application on the fly to detect the resource usage, system stack trace, Hot spot analysis etc.

The idea of malware detection in the run time production platforms poses the challenges of performance degradation to the user accessing the application. Our research study proposes adapting Istio (Istio) framework in analyzing the logs by diverting some production traffic to Intel V-Tune. By this the user will not experience the performance degradation.

Istio's traffic routing rules let you easily control the flow of traffic and API calls between services. Istio simplifies configuration of service-level properties like circuit breakers, timeouts, and retries, and makes it easy to set up important tasks like A/B testing, canary rollouts, and staged rollouts with percentage-based traffic splits. It also provides out-of-box reliability features that help make your application more resilient against failures of dependent services or the network.

Istio's traffic (Traffic shifting) shifting can be configured by two Istio Custom Resources, namely Destination Rule and Virtual Service. In short, in a Destination Rule so-called subsets need to be defined to identify specific versions of a service, and in a Virtual Service different weights can be assigned to these subsets in order to control how much traffic should be directed to each service version.

For the subsets, you'll need to define a Destination Rule like this:

```
kind: DestinationRule
metadata:
  name: xmrig
spec:
  host: xmrig
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
```

Figure 2.1
Istio – Destination Rules

This defines the following subset names: v1, v2 and v3. These subset names can be used later in Virtual Services to route traffic to the appropriate version.

For the weights, it is needed to define a Virtual Service like this:

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: xmrig
spec:
  hosts:
  - movies
  http:
  - route:
    - destination:
        host: xmrig
        subset: v1
      weight: 33
    - destination:
        host: xmrig
        subset: v2
      weight: 33
    - destination:
        host: xmrig
        subset: v3
      weight: 34
```

Figure 2.2
Istio – Virtual Service

Now the Virtual Service is set to route 33% of all traffic to subset v1, 33% to v2 and 34% to v3.

From Kubiya (kubiya.ai), The concept of generative AI describes machine learning algorithms that can create new content from minimal human input. The field has rapidly advanced in the past few years, with projects such as the text authorship tool ChatGPT and realistic image creator DALL-E2 attracting mainstream attention.

Generative AI isn't just for content creators, though. It's also poised to transform technical work in the software engineering and DevOps fields. GitHub Copilot, the controversial "AI pair programmer," is already prompting reconsideration of how code is written, but collaborative AI's potential remains relatively unexplored in the DevOps arena.

Organizations that make regular use of AI are likely to have the best results because their agents will become more adept at anticipating their requirements. However, it's also important not to overreach as you add AI to your workflows. The most successful adoptions will be focused on solving a genuine business need. Assess your processes to identify where bottlenecks exist, such as between dev and ops teams, then target those repetitive use cases with AI.

The solution you select should help you reach your KPIs, such as closing more issues or resolving incidents faster. Otherwise, the AI agent will be underused and could even hinder your natural operating procedures.

Generative AI is one of today’s most quickly maturing technologies. ChatGPT has attained a degree of virality as more researchers, consumers, and organizations begin exploring its capabilities. DALL-E2 has delivered similarly spectacular results, while GitHub Copilot was used by over 1.2 million developers during its first 12 months.

All three technologies demonstrate clear revolutionary potential, but it’s the mixed and highly complex workflows of DevOps that could benefit the most in the long term. DevOps combines the creation of new assets, such as code and configs, with sequential processes like deployment approvals and review requests.

Contrary to some outsider projections, generative AI for DevOps will go beyond mere templating of common file snippets to offer full workflow automation. Using simple conversational phrases, you’ll be able to instruct your agent to take specific actions on your behalf, from provisioning new cloud resources to checking performance in production. The agent will provide a real-time bi-directional feedback loop that improves collaboration, boosts productivity, and reduces the everyday pressures faced by devs.

2.3 Literarture Conclusion

Following environment are considered while evaluating the data.

- Hyperledger Framework is chosen as the case study for Blockchain Deployment.
- The basic software images or binaries will be of Docker images.
- Istio Framework is employed to mitigate the risk of zero downtime in production environment.
- XMRIG Client is used to simulate the miner process.
- Intel V-Tune Profile is used to inspect the miner process.
- Custom Decider Logic is employed to check the logs/reports and conclude the results.
- Repeat the steps from top if necessary for other process.

The primary research method for this study is literature review and the data collected by the Intel V-Tune Profiler in effectively zeroing the threat models. Various results, and the accuracy will be taken into the account as part of the research study. The recent trends show the multifold adaptation of the blockchain technology for all the areas (Finance, Automobile, Healthcare, etc.). The algorithm is fed with data that relates to the problem domain and metadata that attributes a label to the data**.** The Model will look for the system calls and other HW resource utilization (Hot Spot analysis) usages of the existing application deployed in the safe environments before deploying the applications into the cloud platforms. The various literature concepts and gaps (such has minimized the time in detecting the malware and reducing the storage needed to collect the logs of the application) mentioned have well taken into the account in enhancing the research study.

According to Uptycs (2020), With the rise and sky-high valuation of Bitcoin and several other cryptocurrencies, cryptomining-based attacks have continued to dominate the threat landscape. Wormed cyptominer attacks have a greater threshold as they write multiple copies and spread across endpoints in a corporate network. Alongside the mining process, modification of the MSR registers can lead to fatal performance issues of the corporate resources.

Researchers from Uptycs (2020), have revealed that the worm scans for known vulnerabilities in Unix and Linux-based web servers, such as Oracle WebLogic Server or XML-RPC Server.

Following are the points to be considered while evaluating the methods of cryptojacking:

- CVE-2017-11610 is a remote execution flaw used to abuse XML-RPC, while CVE-2020-14882 is a path-traversal vulnerability used for abusing exposed Web Logic servers. It looks like the attackers are attempting to bypass the authorization mechanism by modifying the URL and performing a path traversal (by exploiting CVE-2020-14882) with double encoding on /console/images.
- After exploiting the vulnerabilities, the attack uses a shell script (ldr.sh) that downloads the worm with curl utility. Additionally, the script uses evasion methods such as disabling monitoring agents or altering firewalls.
- The first-stage worm is compiled in Golang and packed with UPX. The worm utilizes a go-bindata package to insert off-the-shelf XMRig cryptominer.

XMRig miners use the RandomX algorithm to produce various unique programs, which are created by data selected from the dataset produced from the hash of a key block.
Memory intensive RandomX programs are executed within a VM. Therefore, the miner disables the hardware prefetcher by abusing the Model Specific Register (MSR), boosting its performance.

While conducting the research, all possible level of hardware support will be explored to scan the malware image and any other software components of platform.

2.4 Summary

The research intends to claim that with the advanced in the computational capacity of the Cloud resources and the efficiency of the Intel V-Tune Profiler together helps to minimize the threats by the malware or hackers’ application in utilizing the HW resources in the K8S environment.

This research study helps in reducing the run time of threat detection of an existing application in the customer environment. The window period of threat detection will be reduced as its makes use of the micro service architecture out of which part of the logs collection will happen in run time.

The K8S is chosen as the reference orchestrating platform to deploy containers. The solution is equally applicable for other orchestrating platforms such as Docker Swarm, Rancher, Red hat OpenShift, Mesos, AWS, Microsoft Azure, GKE etc.

CHAPTER III:
METHODOLOGY

3.1 Introduction

This section will articulate the various aspects of how the research will be conducted, the guiding principles, the nature and philosophy of the research.

There are two main research designs used in academia, quantitative research design and qualitative research design. The quantitative design is primarily about examining the relationship between variables. It involves generating data from samples and analyzing them using statistical techniques and works well with the deductive approach. Qualitative research design, on the other hand, is used with the inductive and abductive approaches. This research design is based on the open-source software tools for analyzing the customer applications for cryptocurrency mining scenario. While quantitative research design involves examining the relationship between variables, qualitative research design involves examining the relationship between entities. The research being presented looks to properly explain the various phenomena around the analysis of the applications in identifying the threat.

The data collection process and verification are summarized as below:

- Install the Intel V-Tune software tool in the host machine.
- Configure the Hot spot analysis and stack trace collection option.
- Identify the customer application crossing the threshold value of CPU usage.
- Start the Hot Spot analysis to generate the required logs, stack trace, LBR etc.
- The duration of the log collection, stack trace depends on the applications running in the customer environment and can be decided by the customer.
- The log collections process results into the data cleaning process that results into the required format (CSV – Comma separated value) for further analysis.
- The Final Decider component of the flow will be the log parser which identify the application is affected by crypto mining logic or not.
- If the parser results in threat, then the further action is decided by the framework suggested by current research to divert the application traffic to another instance without bringing down the running application.
- K8S Istio is the Framework suggested by the research to enable the seamless traffic diversion from the existing affected application to another instance of the unaffected applications.
- The customer environment can be High scale data centre running enterprise applications, small and medium enterprise segment to local desktop applications.
- The Blockchain based De – Centralised applications and the framework components are focus of the research.

3.2 Research Questions

The purpose of this research is to generate a better understanding of the risks involved in the cloud environment with little security involved while deploying the applications. The Hackers will come in various form at various levels to make use of the hardware resources without affecting the applications flow.

The challenge for the customers will be dissect the original applications vs affected applications and the method used to resolve the problem.

- How can the customer identify the applications targeted by crypto miners to make use of the HW platforms?
- Whether the open-source methods/tools like Intel V-Tune profiler can be used to identify?
- Which best resolution methods can be employed to remove the miners targeted application dynamically in the production network?

Furthermore, the research will be critical to increase the TCO for the enterprise customers and for academic purpose to go for an extra mile to tackle the miners of different variants. By conducting this research now, we can expect to see a higher chance of success in identifying the miners.

3.3 Research Design and Strategy

To answer the research questions proposed in the previous section, the researcher will apply quantitative research design. The researcher will use open-source tools and software frameworks to identify the crypto mining process. During the flow, the researcher will note several observations through the course of the logs generated. The data obtained from these logs and with the pipeline structure that logs will traverse through will make an approximate conclusion of the threat. These observations will then be used to formulate the most likely scenario or explanation for the phenomenon.

The investigation undertaken for this study will be of analytics nature. The research aims to explore and helps in exploring more suitable option and fine-tuning methods for the tool to produce more meaningful logs.

The strategy of research is method applied to study the nature of application logs to produce results in-line with the research objectives. Quantitative strategies are best applied to studies that involve mathematical, statistical, and analytical approach.

This research will use a quantitative approach through the means of software tool option exploration and applying the knowledge of the hardware platforms and taking extra mile in checking the specific option for the hardware platform. Specifically, the Intel and AMD processors MSR are values are different in checking the default values.

3.4 Population and Sample

Data samples are needed when the data from the source is extremely large. In our research, the Intel V-Tune tool is feasible enough to generate the data that is considered as the right sample data to analyze. Based on the duration of the customer application monitor, that generate the logs defines the sample size. The sample size consists of the application flow type, repetitive functions, duration of the function execution, CPU usage limit etc.

Some of the advantages of quantitative analysis are:

- Conduct in-depth research: Since quantitative data can be statistically analysed, it is highly likely that the research will be detailed.
- Minimum bias: There are instances in research, where personal bias is involved which leads to incorrect results. Due to the numerical nature of quantitative data, personal bias is reduced to a great extent.
- Accurate results: As the results obtained are objective in nature, they are extremely accurate.

Using quantitative data (Quantitative data) in an investigation is one of the best strategies to guarantee reliable results that allow better decisions. In summary, quantitative data is the basis of statistical analysis. Data that can be measured and verified gives us information about quantities; that is, information that can be measured and written with numbers. Quantitative data defines a number, while qualitative data is descriptive.

For Ex, The Intel V-tune generated data log is considered as Quantitative research data log and it has following pattern:

- CPU Threshold.
- Application function repetitive count.
- Miners pattern count.
- CPU Model specific registers contents.

3.5 Research Instrumentation

For this research, following are the tools employed to ensure that the data being collected for the research was relevant, valuable and could potentially assist in the research.

- PMU (Performance monitoring Units) Profilers.
    - Analysis of Speculative Execution Side Channels.
- Last Branch records.
- Intel v-Tune Profiler.
- Blockchain Hyperledger Fabric.
- Kubernetes Resources (K8S)
- K8S Istio - Mesh Service Framework.

3.5.1 PMU Profilers

The Intel Doc (Document Number: 334467-001, Revision 1.0) refers Performance monitoring units (PMUs) are present in all modern Intel processor generations, allowing profiling utilities to characterize the interaction between software and CPU resources using a rich set of performance metrics. Profilers are critical tools for software to harvest optimal performance out of the CPU hardware.

The programming interfaces that profiling utilities use to access PMUs or related hardware resources consist of:

- A set of instructions (some require privilege access available only in kernel mode, like RDMSR, WRMSR).
- PMU configuration resources: these are typically Model Specific Registers (MSRs).
- Counter register resources: these can include performance counters in the PMU as well as other counter registers accessible as MSRs.

Traditionally, profiling utilities employ special device drivers operating with ring 0 privilege to configure the PMU, access counter registers, and handle interrupts if the profiler supports sampling (i.e. capture samples of incremental data at fine-grain intervals). Some OS, such as Linux, provide API access for root privileged user programs to access privileged resources (such as MSRs). When a user program's profiling needs can be served by counting of events (without the need to capture incremental samples), it is often possible and desirable to implement the profiler as a ring 3 application to make use of these privileged APIs. This simplifies development and deployment of the profiler compared to the traditional approaches of a kernel based driver solution with a command line front-end parser. For security reasons in multi-user OS, the OS only allows access to privileged resources by root users. This implies that the monitoring tool would run with full root rights and have rights to operate privileged resources (as permitted by those API) beyond just monitoring performance events. To configure and use the PMU, read and write accesses to some PMU MSRs are needed by a user-space profiler. However, having full write access to the entire set of MSRs in a CPU can compromise the OS. Thus, full root rights and write access to full set MSRs should be selectively provisioned to a user-space profiler. On secured shared server systems or securely booted clients with secured kernels full MSR access is usually not available.

The goal of this white paper is to define a subset of MSRs and mechanism with the following in mind:

- Writes to the subset of MSRs are to configure performance metric selection and conduct monitoring of the counter registers, without changing any non-PMU states.
- Define write masks that are applicable to the subset of MSRs to ensure the user space profiler operates within the intended monitoring mode (i.e. counting).
- A bridge between the OS-API requirement of full root rights and the desired nonroot permission for user-space applications.
- Allow collecting performance metrics of the whole system, but do not modify any other state. A specialized MSR access layer can then give the monitoring tool only access to this safe “monitoring only” subset of MSRs and allow it to run the monitoring as non-root, without risking compromising the system. Note that

monitoring access is still opt-in by the administrator and cannot be done without an explicit configuration change.

The privilege level gives read and write access to a limited number of MSRs in the logical processor and the physical package. Filtering of input settings specified by the application is written to the MSRs by a privileged software layer (kernel driver or a special secure access layer). The active settings of the MSRs reflect the configuration of the performance monitoring hardware. Input from the non-root application to change any of the secured monitoring registers does not allow:

- Reading or writing any data in memory or in data registers.
- Triggering interrupts.
- Changing state of processes outside the monitoring tool.
- In general, the expectation of performance impact to the target system due to enabling monitoring hardware and the software layer is minimal. Input from the non-root application permit the following changes to the secured monitoring registers:
- Selection of performance monitoring counter events which are supported by the PMU, as well as (optionally) conditioning of performance counter results (e.g. thresholding, edge triggering).
- This includes the ability to monitor events such as cache misses, branch mispredictions and other architectural and micro architectural events

From (Document Number: 336983-004, Revision 4.0), Intel is committed to improving the overall security of computer systems through hardware and software. As detailed by Google Project Zero, https://googleprojectzero.blogspot.com/, a new series of side-channel analysis methods have been discovered that potentially facilitate access to unauthorized information. These methods rely on common properties of both high-performance microprocessors and modern operating systems. Susceptibility to these methods is not limited to Intel processors, nor does it imply the processor is working outside its intended functional specification. All of the methods take advantage of speculative execution, a common technique in processors used to achieve high performance.

3.5.1.1 Speculative Execution

Speculative execution is one of the main techniques used by most modern high-performance processors to improve performance. The concept behind speculative execution is that instructions are executed ahead of knowing that they are required. Without speculative execution, the processor would need to wait for prior instructions to be resolved before executing subsequent ones. By executing instructions speculatively, performance can be increased by minimizing latency and extracting greater parallelism. The results may be discarded if it is discovered that the instructions were not needed after all. The most common form of speculative execution involves the control flow of a program. Instead of waiting for all branch instructions to resolve to determine which operations are needed to execute, the processor predicts the control flow using a highly sophisticated set of mechanisms. Usually the predictions are correct, which allows high performance to be achieved by hiding the latency of the operations that determine the control flow and increasing the parallelism the processor can extract by having a larger pool of instructions to analyze. However, if a prediction is wrong, then the work that was executed speculatively is discarded and the processor will be redirected to execute down the correct instruction path. While speculative operations do not affect the architectural state of the processor, they can affect the microarchitectural state, such as information stored in Translation Lookaside Buffers (TLBs) and caches. The side channel methods described in this white paper take advantage of the fact that the content of caches can be affected by speculative execution.

3.5.1.2 Side Channel Cache Methods

A side channel method works by gaining information through observing the system, such as by measuring microarchitectural properties about the system. Unlike buffer overflows and other vulnerability classes, side channels do not directly influence the execution of the program, nor do they allow data to be modified or deleted. A cache timing side channel involves an agent detecting whether a piece of data is present in a specific level of the processor's caches, where its presence may be used to infer some other piece of information. One method to detect whether the data in question is present is to use timers to measure the latency to access memory at the address. If the memory access takes a short time, then the data must be present in a nearby cache. If the access takes a longer time, then the data may not be in the nearby cache. Google Project Zero identified several methods through which a cache timing side channel can potentially be used to leak secret information.

Variant 1: Bounds Check Bypass:

The bounds check bypass method takes advantage of speculative execution after conditional branch instructions. A malicious actor discovers or causes the creation of "confused deputy" code which allows the attacker to use speculative operations to infer information not normally accessible to the attacker. This method uses speculative operations that occur while the processor is checking whether an input is in bounds, such as checking if the index of an array being read is within acceptable values. It takes advantage of memory accesses to out of bound memory that are performed speculatively before the bounds check resolves. These memory accesses can be used in certain circumstances to leak information to the attacker. If the attacker can identify an appropriate "confused deputy" in a more privileged level, the malicious actor may be able to exploit that deputy to deduce the contents of memory accessible to that deputy but not to the malicious actor. One subvariant of this technique, known as bounds, check bypass store, is to use speculative stores to overwrite younger speculative loads in a way that creates a side channel controlled by a malicious actor.

Refer to the example bounds check bypass store sequence below:

```
 int function(unsigned bound, unsigned long user_key)

{ unsigned long data[8], i;

;; bound is trusted and is never more than 8

i = 0;

while (i < bound) { data[i] = user_key; i++; }

return 0; }
```

It is possible that the above sequence might speculatively overwrite the return address on the stack with user_key. This may allow a malicious actor to specify a key that is actually the RIP of a disclosure gadget that they wish to be speculatively executed.

The steps below describe how an example attack using this method might occur:

- The CPU conditional branch predictor predicts that the loop will iterate 10 iterations, when in reality the loop should have only executed 8 times. After the 10th iteration, the predictor will resolve, fall through, and execute the following instructions. However, the 9th iteration of the loop may speculatively overwrite the return address on the stack.
- The CPU decodes the RET and speculatively fetches instructions based on the prediction in the return stack buffer (RSB). The CPU may speculatively execute those instructions.
- RET executes and redirects the instruction pointer to the value that the RET loaded from memory (which came from the older speculative store of user_key in step 1). The results of any operations speculatively executed in step 2 are discarded.
- The disclosure gadget at the instruction pointer of user_key (which was specified by the malicious actor) speculatively executes and creates a side channel that reveals data specified by the malicious actor.
- The conditional jump that should have ended the loop then executes and redirects the instruction pointer to the next instruction after the loop. This discards the speculative store of user_key that overwrote the return address on the stack, as well as all other operations between step 1 and step 4. 6. The CPU executes the RET again, and the program continues.

Variant 2: Branch Target Injection

The branch target injection method takes advantage of the indirect branch predictors inside the processor that are used to direct what operations are speculatively executed. By influencing how the indirect branch predictors operate, an attacker can cause malicious code to be speculatively executed and then use the effects such code has on the caches to infer data values. For conditional direct branches, there are only two options as to what code speculatively executes –either the target of the branch or the fall-through path of instructions directly subsequent to the branch. The attacker cannot cause code to be speculatively executed outside of those locations. Indirect branches, however, can cause speculative execution of code at a wider set of targets. This method works by causing an indirect branch to speculatively execute a 'gadget' which creates a side channel based on sensitive data available to the victim. The ability to interfere with the processor's predictors to cause such a side channel is highly dependent on the microarchitectural implementation, and the exact methods used may thus vary across different processor families and generations. For example, the indirect branch predictors in some processor implementations may only use a subset of the overall address to index into the predictor. If an attacker can discern what subset of bits are used, the attacker can use this information to create interference due to aliasing. Similarly, on processors that support Intel® Hyperthreading Technology (Intel® HT Technology), whether one thread's behavior can influence the prediction of the other thread is a consideration. The branch target injection method can only occur for a near indirect branch instruction.

Variant 3: Rogue Data Cache Load

The rogue data cache load method involves an application (user) attacker directly probing kernel (supervisor) memory. Such an operation would typically result in a program error (page fault due to page table permissions). However, it is possible for such an operation to be speculatively executed under certain conditions for certain implementations. For instance, on some implementations such a speculative operation will only pass data on to subsequent operations if the data is resident in the lowest level data cache (L1). This can allow the data in question to be queried by the application, leading to a side channel that reveals supervisor data. This method only applies to regions of memory designated supervisor-only by the page tables; not memory designated as not present.

The rogue system register read method, as described as Variant 3a in the ARM* whitepaper1, uses both speculative execution and side channel cache methods to infer the value of some processor system register state which is not architecturally accessible by the attacker. This method uses speculative execution of instructions that read system register state while the processor is operating at a mode or privilege level that does not architecturally allow the reading of that state. The set of system registers that can have their value inferred by this method is implementation specific. Although these operations will architecturally fault or VM exit, in certain cases, they may return data accessible to subsequent instructions in the speculative execution path. These subsequent instructions can then create a side channel to infer the system register state. Intel's analysis is that the majority of state exposed by the Variant 3a method is not secret or sensitive, nor directly enables attack or exposure of user data. The use of the Variant 3a method by an attacker may result in the exposure of the physical addresses for some data structures and may also expose the linear addresses of some kernel software entry points. Knowledge of these physical and linear addresses may enable attackers to determine the addresses of other kernel data and code elements, which may impact the efficacy of the Kernel Address Space Layout Randomization (KASLR) technique. KASLR, as a security defense in-depth feature, has been subject to a number of attacks in recent years; in particular against local attackers who can control code execution. As the rogue system register read method involves attacker-controlled code execution, a local attacker employing rogue system register read to break KASLR may be low impact for most end users.

Variant 4: Speculative Store Bypass

The speculative store bypass method takes advantage of a performance feature present in many high-performance processors that allows loads to speculatively execute even if the address of preceding potentially overlapping store is unknown. In such a case, this may allow a load to speculatively read a stale data value. The processor will eventually correct such cases, but an attacker may be able to discover “confused deputy” code which may allow them to use speculative execution to reveal the value of memory that is not normally accessible to them. In a language-based security environment (e.g., a managed runtime), where an attacker is able to influence the generation of code, an attacker may be able to create such a confused deputy. Intel has not currently observed this method in situations where the attacker has to discover such a confused deputy instead of being able to cause it to be generated.

3.5.2 Last Branch Records

Support of the architectural LBR feature in a logical processor is reported in CPUID.(EAX=07H, ECX=0):EDX[19]=1. When the architectural LBR feature is supported, capability details like the number of LBR records that are available is indicated in CPUID.(EAX=1CH, ECX=0):EAX[7:0]. The number of LBR records available varies across processor generations, so software is expected to query the CPUID.(EAX=1CH, ECX=0):EAX[7:0] reported value and only access the available LBR records.

Last Branch Records (LBRs) enable recording of software path history by logging taken branches and other control flow transfers within processor registers. Each LBR record or entry is comprised of three MSRs:

- IA32_LBR_x_FROM_IP − Holds the source IP of the operation.
- IA32_LBR_x_TO_IP − Holds the destination IP of the operation.
- IA32_LBR_x_INFO − Holds metadata for the operation, including mispredict, TSX, and elapsed cycle time information.

LBR records are stored in age order. The most recent LBR entry is stored in IA32_LBR_0_*, the next youngest in IA32_LBR_1_*, and so on. When an operation to be recorded completes (retires) with LBRs enabled (IA32_LBR_CTL.LBREn=1), older LBR entries are shifted in the LBR array by one entry, then a record of the new operation is written into entry 0. The number of LBR entries available for recording operations is dictated by the value in IA32_LBR_DEPTH.DEPTH. By default, the DEPTH value matches the maximum number of LBRs supported by the processor, but software may opt to use fewer in order to achieve reduced context switch latency. In addition to the LBRs, there is a single Last Event Record (LER). It records the last taken branch preceding the last exception, hardware interrupt, or software interrupt. Like LBRs, the LER is comprised of three MSRs (IA32_LER_FROM_IP, IA32_LER_TO_IP, IA32_LER_INFO), and is subject to the same dependencies on enabling and filtering.

LBRs can log most control flow transfer operations. The source IP recorded for a branch instruction is the IP of that instruction. For events that take place between instructions, the source IP recorded is the IP of the next sequential instruction. The destination IP recorded is always the target of the branch or event, the next instruction that will execute.

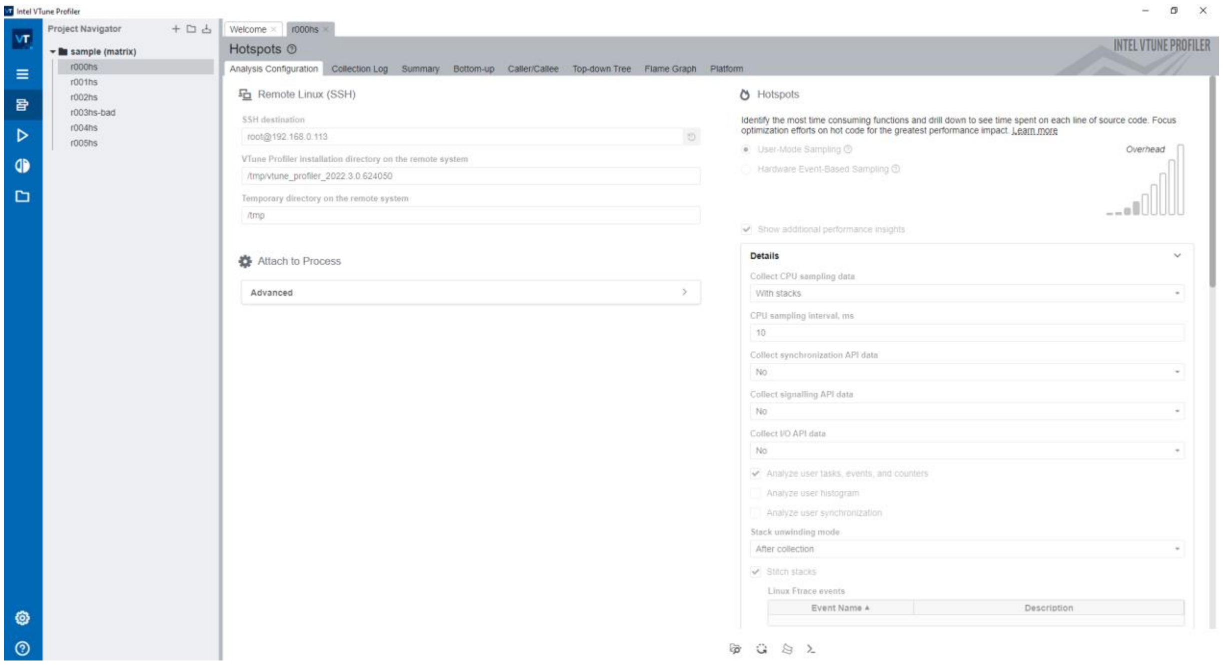
3.5.3 Intel V-Tune Profiler

Intel V-Tune profiler tool is used to locate or determine:

- The most time-consuming (hot) functions in your application and/or on the whole system.
- Sections of code that do not effectively utilize available processor time.
- The best sections of code to optimize for sequential performance and for threaded performance.
- Synchronization objects that affect the application performance.
- Whether, where, and why your application spends time on input/output operations.
- Whether your application is CPU or GPU bound and how effectively it offloads code to the GPU.
- The performance impact of different synchronization methods, different numbers of threads, or different Algorithms.
- Thread activity and transitions
- Hardware-related issues in your code such as data sharing, cache misses, branch misprediction, and others.

Step 1: Configure the Intel V-Tune profiler tool :

Following figure depicts the main screen of the tool



Remote Linux (SSH)

SSH destination: root@192.168.0.113

VTune Profiler installation directory on the remote system: /tmp/vtune_profiler_2022.3.0.624050

Temporary directory on the remote system: /tmp

Attach to Process

Advanced

Hotspots

Identify the most time consuming functions and drill down to see time spent on each line of source code. Focus optimization efforts on hot code for the greatest performance impact. Learn more

● User-Mode Sampling

○ Hardware Event-Based Sampling

✓ Show additional performance insights

Details

Collect CPU sampling data: With stacks

CPU sampling interval, ms: 10

Collect synchronization API data: No

Collect signalling API data: No

Collect I/O API data: No

✓ Analyze user tasks, events, and counters

Analyze user histogram

Analyze user synchronization

Stack unwinding mode: After collection

✓ Stitch stacks

Linux Ftrace events

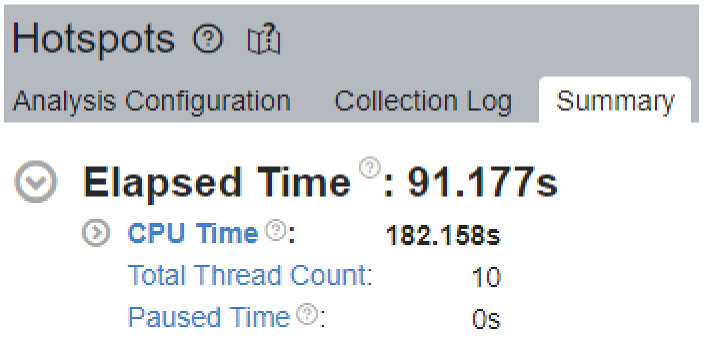
Event Name ▲	Description

Figure 3.1
Intel V-Tune Profiler

Intel V-Tune Profiler profiles the given application and collects data. Once this process completes, Intel V-Tune Profiler finalizes the collected results and resolves symbol information.

Step 2: Elapsed Time (Duration of the application to run)

The Hotspots section of the application depicted below emphasizes the CPU threshold value.



Hotspots

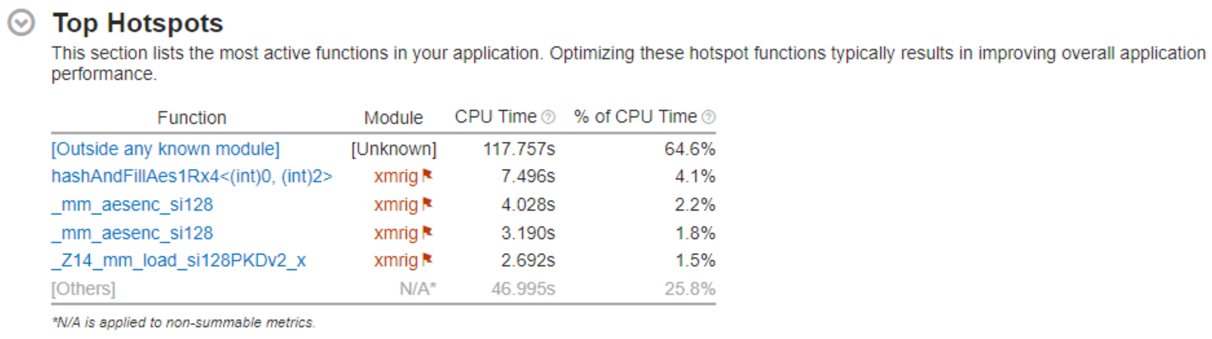
Elapsed Time: 91.177s

CPU Time:	182.158s
Total Thread Count:	10
Paused Time:	0s

Figure 3.2
Intel V-Tune Profile – ELP Time

Step 3: The application most active functions

Following figure gives the glimpse of the application call flow and CPU usage time



Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

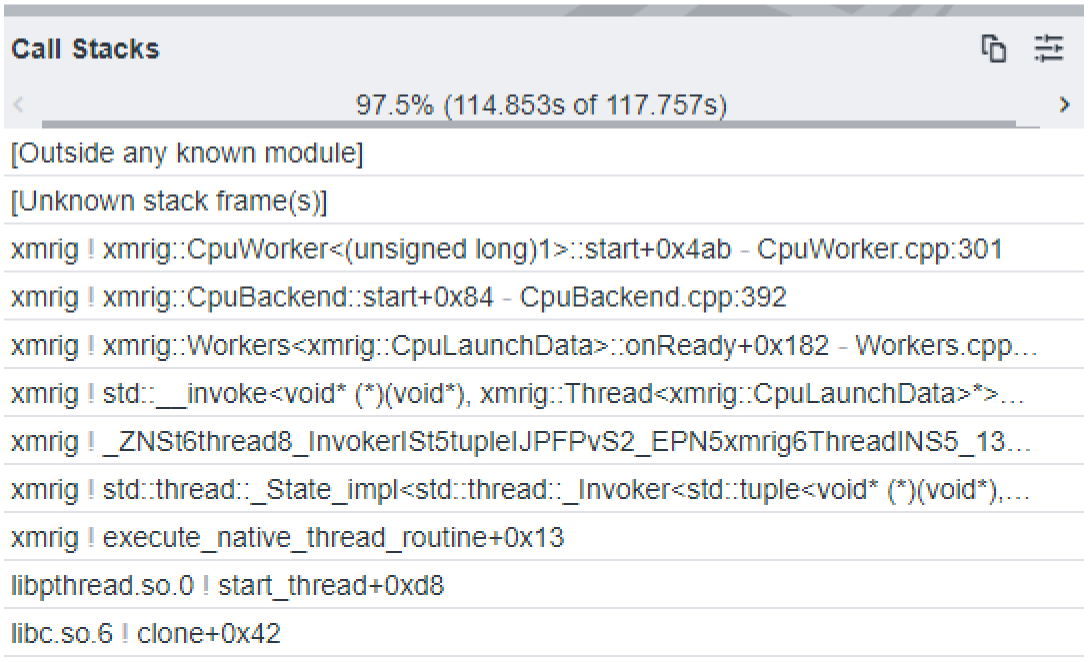
Function	Module	CPU Time	% of CPU Time
[Outside any known module]	[Unknown]	117.757s	64.6%
hashAndFillAes1Rx4<(int)0, (int)2>	xmrig	7.496s	4.1%
_mm_aesenc_si128	xmrig	4.028s	2.2%
_mm_aesenc_si128	xmrig	3.190s	1.8%
_Z14_mm_load_si128PKDv2_x	xmrig	2.692s	1.5%
[Others]	N/A*	46.995s	25.8%

*N/A is applied to non-summable metrics.

Figure 3.3
Intel V-Tune Profile – Hotspots

Step 4: Call Stacks trace

The stack trace of any applications is on the vital info in understanding the application flow as follows:



```
Call Stacks
97.5% (114.853s of 117.757s)
[Outside any known module]
[Unknown stack frame(s)]
xmrig ! xmrig::CpuWorker<(unsigned long)1>::start+0x4ab - CpuWorker.cpp:301
xmrig ! xmrig::CpuBackend::start+0x84 - CpuBackend.cpp:392
xmrig ! xmrig::Workers<xmrig::CpuLaunchData>::onReady+0x182 - Workers.cpp...
xmrig ! std::__invoke<void* (*)(void*), xmrig::Thread<xmrig::CpuLaunchData>*>...
xmrig ! _ZNSt6thread8_InvokerISt5tupleIJPFPvS2_EPN5xmrig6ThreadINS5_13...
xmrig ! std::thread::_State_impl<std::thread::_Invoker<std::tuple<void* (*)(void*),...
xmrig ! execute_native_thread_routine+0x13
libpthread.so.0 ! start_thread+0xd8
libc.so.6 ! clone+0x42
```

Figure 3.4
Intel V-Tune Profile – Stack Trace

Step 5: Flame Graph

The flame graph is a graphical representation of the data contained in the tabular Top-Down view

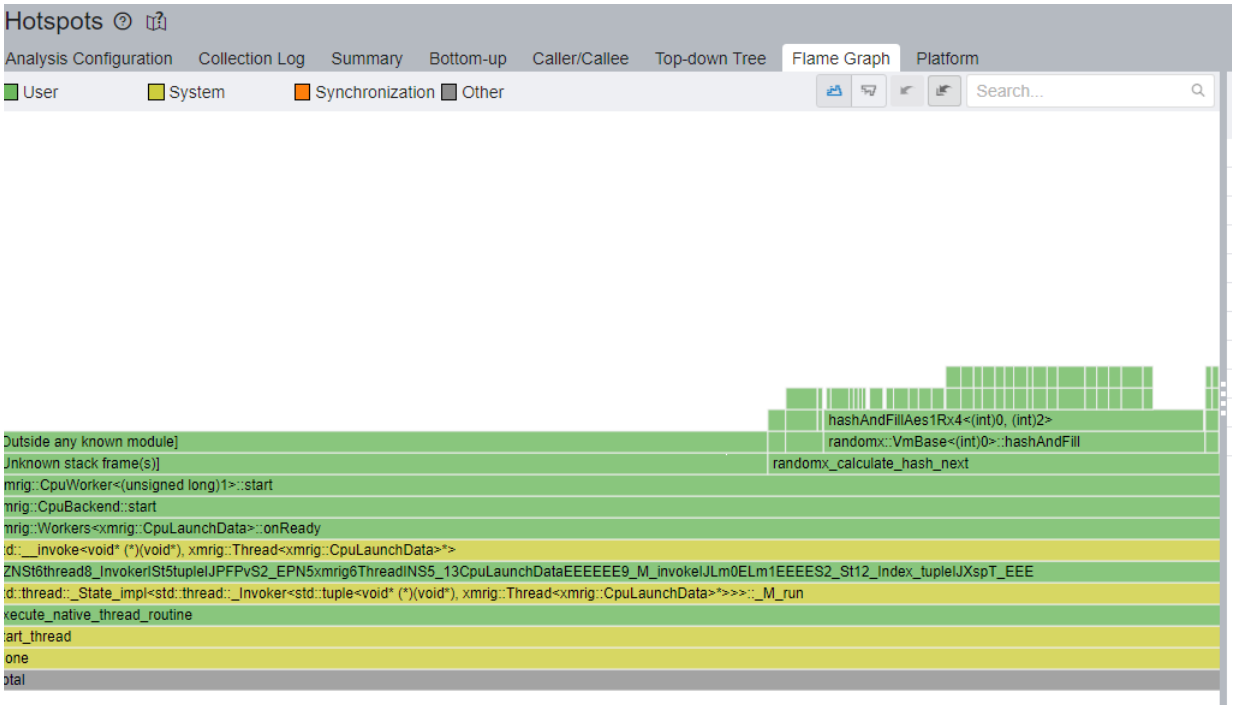

Hotspots
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Flame Graph
Platform
🟩 User
🟨 System
🟧 Synchronization
⬛ Other
Search...
hashAndFillAes1Rx4<(int)0, (int)2>
Dutside any known module]
randomx::VmBase<(int)0>::hashAndFill
Jnknown stack frame(s)]
randomx_calculate_hash_next
mrig::CpuWorker<(unsigned long)1>::start
nrig::CpuBackend::start
nrig::Workers<xmrig::CpuLaunchData>::onReady
:d::__invoke<void* (*)(void*), xmrig::Thread<xmrig::CpuLaunchData>*>
ZNSt6thread8_InvokerISt5tupleIJPFPvS2_EPN5xmrig6ThreadINS5_13CpuLaunchDataEEEEEE9_M_invokeIJLm0ELm1EEEES2_St12_Index_tupleIJXspT_EEE
:d::thread::_State_impl<std::thread::_Invoker<std::tuple<void* (*)(void*), xmrig::Thread<xmrig::CpuLaunchData>*>>>::_M_run
xecute_native_thread_routine
:art_thread
one
otal

Figure 3.5
Intel V-Tune Profile – Flame Graph

A Flame Graph is a visual representation of the stacks and stack frames in your application. Every box in the graph represents a stack frame with the complete function name. The horizontal axis shows the stack profile population, sorted alphabetically. The vertical axis shows the stack depth, starting from zero at the bottom.

This graph displays stacks and stack frames executing application. Every box in the graph represents a stack frame with the complete function name. The horizontal axis shows the stack profile population, sorted alphabetically. The vertical axis shows the stack depth, starting from zero at the bottom.

The flame graph does not display data over time. The width of each box in the graph indicates the percentage of the function CPU time to total CPU time. The total function time includes processing times of the function and all its children (callees).

The flame graph is a graphical representation of the data contained in the tabular Top-Down view.

Following are the some more features that makes Intel V-Tune profiler as the go to tool in the developer community.

- Examine how efficiently the code is threaded. Identify threading issues that impact performance.
- Evaluate compute-intensive or throughput HPC applications for efficient CPU use, vectorization, and memory use.
- Locate performance bottlenecks in I/O-intensive applications. Explore how effectively the hardware processes I/O traffic generated by external PCIe* devices or integrated accelerators.
- See a holistic view of system behavior for long-running workloads with Platform Profiler.
- Get a fine-grained overview for short-running workloads with System Overview.
- Locate hot spots—the most time-consuming parts of your code.
- Visualize hot code paths and time spent in each function and with its callees with Flame Graph.
- Identify the most significant hardware issues that affect the performance of your application with microarchitecture exploration analysis.

- Pinpoint memory-access-related issues such as cache misses and high-bandwidth problems.
- Optimize GPU offload schema and data transfers for SYCL, OpenCL code, Microsoft DirectX*, or OpenMP* offload code. Identify the most time-consuming GPU kernels for further optimization.
- Analyze GPU-bound code for performance bottlenecks caused by microarchitectural constraints or inefficient kernel algorithms.
- Explore CPU and FPGA interactions, and FPGA use.
- Characterize performance aspects of large-scale message passing interface (MPI) and OpenMP workloads.
- Identify scalability issues and get recommendations for in-depth analysis.

3.5.4 Hyperledger Fabric

Hyperledger Fabric is an enterprise-grade, distributed ledger platform that offers modularity and versatility for a broad set of industry use cases. The modular architecture for Hyperledger Fabric accommodates the diversity of enterprise use cases through plug and play components, such as consensus, privacy, and membership services.

- Private: membership to channels is controlled
- Permissioned: members are identifiable via PKI
- Decentralized: full ledger replicated to members
- Ledger is an append-only blockchain
- Cryptography ensures an immutable ledger, i.e., a system-of-proof
- Smart contracts support queries.
- Consensus: agreeing valid transactions and applying them in order

At its core, blockchain is a new type of data system that maintains and records data in a way that allows multiple stakeholders to confidently share access to the same data and information. A blockchain is a type of Distributed Ledger Technology, meaning it is a data ledger that is shared by multiple entities operating on a distributed network.

This technology operates by recording and storing every transaction across the network in a cryptographically linked block structure that is replicated across network participants. Every time a new data block is created, it is appended to the end of the existing chain formed by all previous transactions, thus creating a chain of blocks called the blockchain.

This blockchain (Hyperledger Fabric Components) format contains records of all transactions and data, starting from the inception of that data structure.

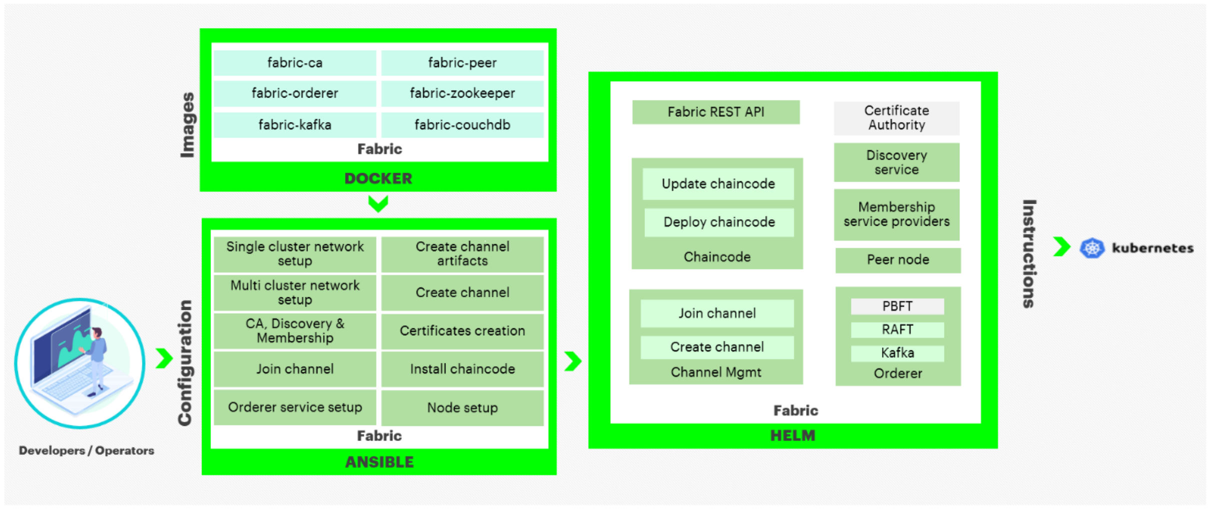


Figure 3.6 Hyperledger Topology

Hyperledger uses officially published Hyperledger Fabric Docker images from hub.docker.com.

The following are the Hyperledger Fabric Docker Images that are targeted by the hackers in Hyperledger framework.

- fabric-ca - Hyperledger Fabric Certificate Authority
- fabric-couchdb - CouchDB for Hyperledger Fabric Peer
- fabric-kafka - Kafka for Hyperledger Fabric Orderer
- fabric-orderer - Hyperledger Fabric Orderer
- fabric-peer - Hyperledger Fabric Peer
- fabric-zookeeper - Zookeeper for Hyperledger Fabric Orderer

3.5.4.1 Key Concepts of Hyperledger Fabric

According to Hyperledger Fabric(hyperledger-fabric.readthedocs.io, 2023), it is private and permissioned. Rather than an open permissionless system that allows unknown identities to participate in the network (requiring protocols like “proof of work” to validate transactions and secure the network), the members of a Hyperledger Fabric network enroll through a trusted Membership Service Provider (MSP).

Hyperledger Fabric also offers several pluggable options. Ledger data can be stored in multiple formats, consensus mechanisms can be swapped in and out, and different MSPs are supported.

Hyperledger Fabric also offers the ability to create channels, allowing a group of participants to create a separate ledger of transactions. This is an especially important option for networks where some participants might be competitors and not want every transaction they make — a special price they’re offering to some participants and not others, for example — known to every participant. If two participants form a channel, then those participants — and no others — have copies of the ledger for that channel.

- Shared Ledger

  Hyperledger Fabric has a ledger subsystem comprising two components: the world state and the transaction log. Each participant has a copy of the ledger to every Hyperledger Fabric network they belong to.

  The world state component describes the state of the ledger at a given point in time. It’s the database of the ledger. The transaction log component records all transactions which have resulted in the current value of the world state; it’s the update history for the world state. The ledger, then, is a combination of the world state database and the transaction log history.

  The ledger has a replaceable data store for the world state. By default, this is a LevelDB key-value store database. The transaction log does not need to be pluggable. It simply records the before and after values of the ledger database being used by the blockchain network.

- Smart Contracts

  Hyperledger Fabric smart contracts are written in chaincode and are invoked by an application external to the blockchain when that application needs to interact with the ledger. In most cases, chaincode interacts only with the database component of the ledger, the world state (querying it, for example), and not the transaction log.

  Chaincode can be implemented in several programming languages. Currently, Go, Node.js, and Java chaincode are supported.

- Privacy

  Depending on the needs of a network, participants in a Business-to-Business (B2B) network might be extremely sensitive about how much information they share. For other networks, privacy will not be a top concern.

  Hyperledger Fabric supports networks where privacy (using channels) is a key operational requirement as well as networks that are comparatively open.

- Consensus

  Transactions must be written to the ledger in the order in which they occur, even though they might be between different sets of participants within the network. For this to happen, the order of transactions must be established and a method for rejecting bad transactions that have been inserted into the ledger in error (or maliciously) must be put into place.

  This is a thoroughly researched area of computer science, and there are many ways to achieve it, each with different trade-offs. For example, PBFT (Practical Byzantine Fault Tolerance) can provide a mechanism for file replicas to communicate with each other to keep each copy consistent, even in the event of corruption. Alternatively, in Bitcoin, ordering happens through a process called mining where competing computers race to solve a cryptographic puzzle which defines the order that all processes subsequently build upon.

  Hyperledger Fabric has been designed to allow network starters to choose a consensus mechanism that best represents the relationships that exist between participants. As with privacy, there is a spectrum of needs; from networks that are highly structured in their relationships to those that are more peer-to-peer.

3.5.4.2 Hyperledger Fabric Model

This section outlines the key design features woven into Hyperledger Fabric that fulfill its promise of a comprehensive, yet customizable, enterprise blockchain solution:

- Assets — Asset definitions enable the exchange of almost anything with monetary value over the network, from whole foods to antique cars to currency futures.
  Assets can range from the tangible (real estate and hardware) to the intangible (contracts and intellectual property). Hyperledger Fabric provides the ability to modify assets using chaincode transactions.

  Assets are represented in Hyperledger Fabric as a collection of key-value pairs, with state changes recorded as transactions on a Channel ledger. Assets can be represented in binary and/or JSON form.
- Chaincode — Chaincode execution is partitioned from transaction ordering, limiting the required levels of trust and verification across node types, and optimizing network scalability and performance.
  Chaincode is software defining an asset or assets, and the transaction instructions for modifying the asset(s); in other words, it's the business logic. Chaincode enforces the rules for reading or altering key-value pairs or other state database information. Chaincode functions execute against the ledger's current state database and are initiated through a transaction proposal. Chaincode execution results in a set of key-value writes (write set) that can be submitted to the network and applied to the ledger on all peers.

- Ledger Features — The immutable, shared ledger encodes the entire transaction history for each channel and includes SQL-like query capability for efficient auditing and dispute resolution.

  The ledger is the sequenced, tamper-resistant record of all state transitions in the fabric. State transitions are a result of chaincode invocations ('transactions') submitted by participating parties. Each transaction results in a set of asset key-value pairs that are committed to the ledger as creates, updates, or deletes.

  The ledger is comprised of a blockchain ('chain') to store the immutable, sequenced record in blocks, as well as a state database to maintain current fabric state. There is one ledger per channel. Each peer maintains a copy of the ledger for each channel of which they are a member.

  - ✓ Query and update ledger using key-based lookups, range queries, and composite key queries.
  - ✓ Read-only queries using a rich query language (if using CouchDB as state database).
  - ✓ Read-only history queries — Query ledger history for a key, enabling data provenance scenarios.
  - ✓ Transactions consist of the versions of keys/values that were read in chaincode (read set) and keys/values that were written in chaincode (write set).

- ✓ Transactions contain signatures of every endorsing peer and are submitted to ordering service.
- ✓ Transactions are ordered into blocks and are “delivered” from an ordering service to peers on a channel.
- ✓ Peers validate transactions against endorsement policies and enforce the policies.
- ✓ Prior to appending a block, a versioning check is performed to ensure that states for assets that were read have not changed since chaincode execution time.
- ✓ There is immutability once a transaction is validated and committed
- ✓ A channel’s ledger contains a configuration block defining policies, access control lists, and other pertinent information.
- ✓ Channels contain Membership Service Provider instances allowing for crypto materials to be derived from different certificate authorities.

- Privacy — Channels and private data collections enable private and confidential multi-lateral transactions that are usually required by competing businesses and regulated industries that exchange assets on a common network.

  Hyperledger Fabric employs an immutable ledger on a per-channel basis, as well as chaincode that can manipulate and modify the current state of assets (i.e. update key-value pairs). A ledger exists in the scope of a channel — it can be shared across the entire network (assuming every participant is operating on one common channel) — or it can be privatized to include only a specific set of participants.

  In the latter scenario, these participants would create a separate channel and thereby isolate/segregate their transactions and ledger. In order to solve scenarios that want to bridge the gap between total transparency and privacy, chaincode can be installed only on peers that need to access the asset states to perform reads and writes (in other words, if a chaincode is not installed on a peer, it will not be able to properly interface with the ledger).

  When a subset of organizations on that channel need to keep their transaction data confidential, a private data collection (collection) is used to segregate this data in a private database, logically separate from the channel ledger, accessible only to the authorized subset of organizations.

Thus, channels keep transactions private from the broader network whereas collections keep data private between subsets of organizations on the channel.

To further obfuscate the data, values within chaincode can be encrypted (in part or in total) using common cryptographic algorithms such as AES before sending transactions to the ordering service and appending blocks to the ledger. Once encrypted data has been written to the ledger, it can be decrypted only by a user in possession of the corresponding key that was used to generate the cipher text.

- Security & Membership Services — Permissioned membership provides a trusted blockchain network, where participants know that all transactions can be detected and traced by authorized regulators and auditors.

  Hyperledger Fabric underpins a transactional network where all participants have known identities. Public Key Infrastructure is used to generate cryptographic certificates which are tied to organizations, network components, and end users or client applications. As a result, data access control can be manipulated and governed on the broader network and on channel levels. This "permissioned" notion of Hyperledger Fabric, coupled with the existence and capabilities of channels, helps address scenarios where privacy and confidentiality are paramount concerns.

- Consensus — A unique approach to consensus enables the flexibility and scalability needed for the enterprise.

  In distributed ledger technology, consensus has recently become synonymous with a specific algorithm, within a single function. However, consensus encompasses more than simply agreeing upon the order of transactions, and this differentiation is highlighted in Hyperledger Fabric through its fundamental role in the entire transaction flow, from proposal and endorsement, to ordering, validation and commitment. In a nutshell, consensus is defined as the full-circle verification of the correctness of a set of transactions comprising a block.

  Consensus is achieved ultimately when the order and results of a block's transactions have met the explicit policy criteria checks. These checks and balances take place during the lifecycle of a transaction, and include the usage of endorsement policies to dictate which specific members must endorse a certain transaction class, as well as system chain codes to ensure that these policies are enforced and upheld. Prior to commitment, the peers will employ these system chaincodes to make sure that enough endorsements are present, and that they were derived from the appropriate entities. Moreover, a versioning check will take place during which the current state of the ledger is agreed or consented upon, before any blocks containing transactions are appended to the ledger. This final check provides protection against double spend operations and other threats that might compromise data integrity and allows for functions to be executed against non-static variables.

In addition to the multitude of endorsement, validity and versioning checks that take place, there are also ongoing identity verifications happening in all directions of the transaction flow. Access control lists are implemented on hierarchical layers of the network (ordering service down to channels), and payloads are repeatedly signed, verified, and authenticated as a transaction proposal passes through the different architectural components. To conclude, consensus is not merely limited to the agreed upon order of a batch of transactions; rather, it is an overarching characterization that is achieved as a by-product of the ongoing verifications that take place during a transaction’s journey from proposal to commitment

3.5.5 Kubernetes Resources (K8S)

Kubernetes (Kubernetes.io) is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Traditional deployment era:

Early on, organizations ran applications on physical servers. There was no way to define resource boundaries for applications in a physical server, and this caused resource allocation issues. For example, if multiple applications run on a physical server, there can be instances where one application would take up most of the resources, and as a result, the other applications would underperform. A solution for this would be to run each application on a different physical server. But this did not scale as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

Virtualized deployment era:

As a solution, virtualization was introduced. It allows you to run multiple Virtual Machines (VMs) on a single physical server's CPU. Virtualization allows applications to be isolated between VMs and provides a level of security as the information of one application cannot be freely accessed by another application.

Virtualization allows better utilization of resources in a physical server and allows better scalability because an application can be added or updated easily, reduces hardware costs, and much more. With virtualization you can present a set of physical resources as a cluster of disposable virtual machines.

Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

Container deployment era:

Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications. Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, share of CPU, memory, process space, and more. As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.

Containers have become popular because they provide extra benefits, such as:

- Agile application creation and deployment: increased ease and efficiency of container image creation compared to VM image use.
- Continuous development, integration, and deployment provides for reliable and frequent container image build and deployment with quick and efficient rollbacks (due to image immutability).
- Dev and Ops separation of concerns: create application container images at build/release time rather than deployment time, thereby decoupling applications from infrastructure.
- Observability: not only surfaces OS-level information and metrics, but also application health and other signals.
- Environmental consistency across development, testing, and production: runs the same on a laptop as it does in the cloud.
- Cloud and OS distribution portability: runs on Ubuntu, RHEL, CoreOS, on-premises, on major public clouds, and anywhere else.
- Application-centric management raises the level of abstraction from running an OS on virtual hardware to running an application on an OS using logical resources.
- Loosely coupled, distributed, elastic, liberated micro-services: applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a monolithic stack running on one big single-purpose machine.
- Resource isolation: predictable application performance.
- Resource utilization: high efficiency and density

Kubernetes provides you with:

- Service discovery and load balancing Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- Storage orchestration Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- Automated rollouts and rollbacks You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- Automatic bin packing You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- Self-healing Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- Secret and configuration management Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can

deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

3.5.5.1 Kubernetes Components

A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

Control Plane Components

- **kube-apiserver**

  The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

  The main implementation of a Kubernetes API server is kube-apiserver. kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

- **Etcd**

  Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

  If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for the data.

- **kube-scheduler**

  Control plane component that watches for newly created Pods with no assigned node and selects a node for them to run on.

  Factors considered for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

- **kube-controller-manager**

  Control plane component that runs controller processes.

  Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

  Some types of these controllers are:

  - ✓ Node controller: Responsible for noticing and responding when nodes go down.
  - ✓ Job controller: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
  - ✓ EndpointSlice controller: Populates EndpointSlice objects (to provide a link between Services and Pods).
  - ✓ ServiceAccount controller: Create default ServiceAccounts for new namespaces.

- **cloud-controller-manager**

  A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.

The cloud-controller-manager only runs controllers that are specific to your cloud provider. If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.

As with the kube-controller-manager, the cloud-controller-manager combines several logically independent control loops into a single binary that you run as a single process. You can scale horizontally (run more than one copy) to improve performance or to help tolerate failures.

The following controllers can have cloud provider dependencies:

Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding

Route controller: For setting up routes in the underlying cloud infrastructure

Service controller: For creating, updating and deleting cloud provider load balancers

Node Components

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

- **Kubelet**

  An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

  The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

- **kube-proxy**

  kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

  kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

  kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

- **Container runtime**

  The container runtime is the software that is responsible for running containers.

  Kubernetes supports container runtimes such as containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface).

- **Addons**

  Addons use Kubernetes resources (DaemonSet, Deployment, etc) to implement cluster features. Because these are providing cluster-level features, namespaced resources for addons belong within the kube-system namespace.

- **DNS**

  While the other addons are not strictly required, all Kubernetes clusters should have cluster DNS, as many examples rely on it.

  Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services.

  Containers started by Kubernetes automatically include this DNS server in their DNS searches.

- **Web UI (Dashboard)**

  Dashboard is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage and troubleshoot applications running in the cluster, as well as the cluster itself.

- **Container Resource Monitoring**

  Container Resource Monitoring records generic time-series metrics about containers in a central database, and provides a UI for browsing that data.

- **Cluster-level Logging**

  A cluster-level logging mechanism is responsible for saving container logs to a central log store with search/browsing interface.

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine.

Resource Management for Pods and Containers

One can specify how much of each resource a container needs. The most common resources to specify are CPU and memory (RAM); there are others.

When you specify the resource request for containers in a Pod, the kube-scheduler uses this information to decide which node to place the Pod on. When you specify a resource limit for a container, the kubelet enforces those limits so that the running container is not allowed to use more of that resource than the limit you set. The kubelet also reserves at least the request amount of that system resource specifically for that container to use.

If the node where a Pod is running has enough of a resource available, it's possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit.

For example, if you set a memory request of 256 MiB for a container, and that container is in a Pod scheduled to a Node with 8GiB of memory and no other Pods, then the container can try to use more RAM.

If you set a memory limit of 4GiB for that container, the kubelet (and container runtime) enforce the limit. The runtime prevents the container from using more than the configured resource limit. For example: when a process in the container tries to consume more than the allowed amount of memory, the system kernel terminates the process that attempted the allocation, with an out of memory (OOM) error.

Limits can be implemented either reactively (the system intervenes once it sees a violation) or by enforcement (the system prevents the container from ever exceeding the limit). Different runtimes can have different ways to implement the same restrictions.

Resource types

CPU and memory are each a resource type. A resource type has a base unit. CPU represents compute processing and is specified in units of Kubernetes CPUs. Memory is specified in units of bytes. For Linux workloads, you can specify huge page resources. Huge pages are a Linux-specific feature where the node kernel allocates blocks of memory that are much larger than the default page size.

For example, on a system where the default page size is 4KiB, you could specify a limit, hugepages-2Mi: 80Mi. If the container tries allocating over 40 2MiB huge pages (a total of 80 MiB), that allocation fails.

CPU and memory are collectively referred to as compute resources, or resources. Compute resources are measurable quantities that can be requested, allocated, and consumed. They are distinct from API resources. API resources, such as Pods and Services are objects that can be read and modified through the Kubernetes API server.

For each container, you can specify resource limits and requests, including the following:

- ✓ spec.containers[].resources.limits.cpu
- ✓ spec.containers[].resources.limits.memory
- ✓ spec.containers[].resources.limits.hugepages-<size>
- ✓ spec.containers[].resources.requests.cpu
- ✓ spec.containers[].resources.requests.memory
- ✓ spec.containers[].resources.requests.hugepages-<size>

Although you can only specify requests and limits for individual containers, it is also useful to think about the overall resource requests and limits for a Pod. For a particular resource, a Pod resource request/limit is the sum of the resource requests/limits of that type for each container in the Pod.

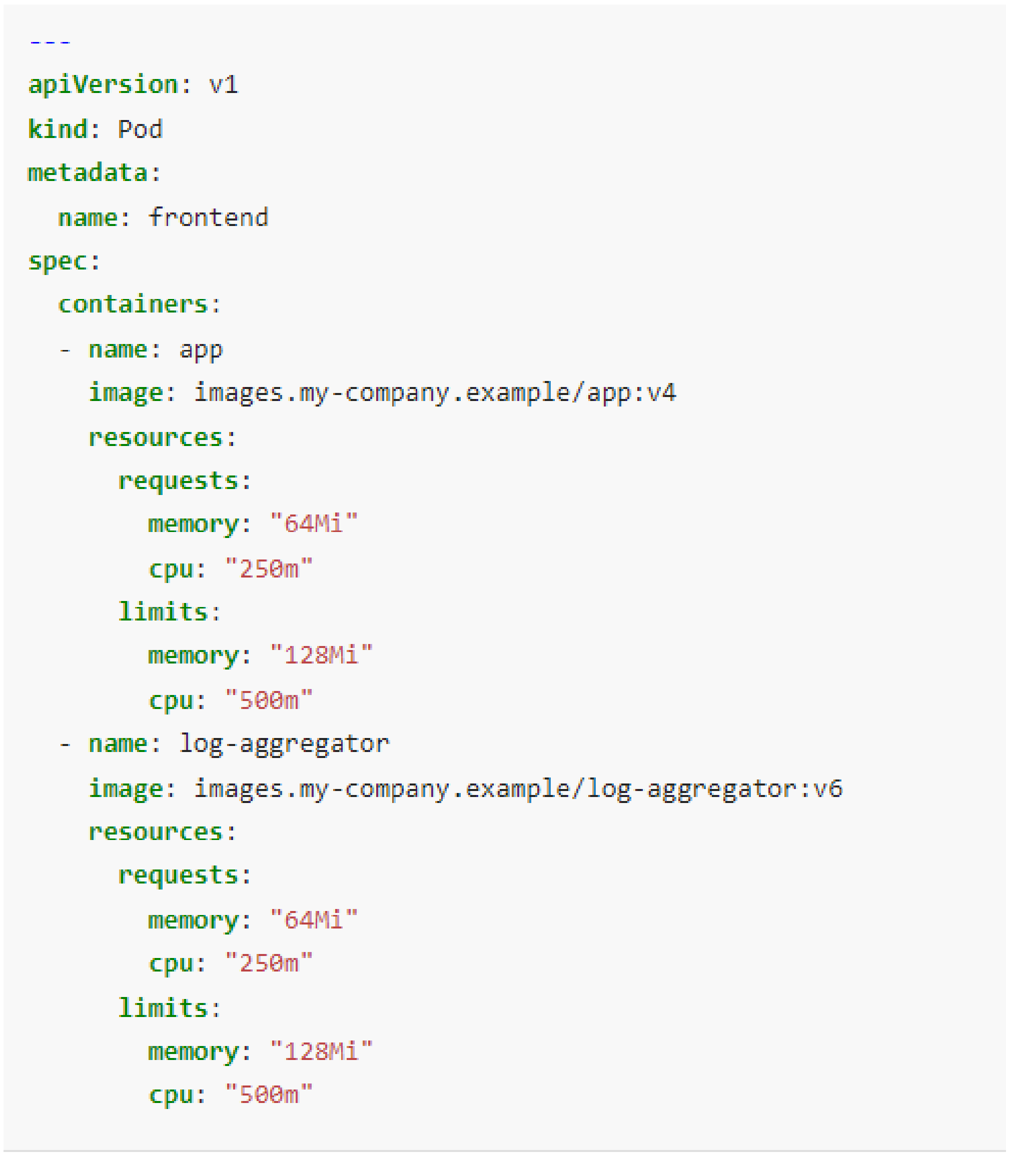
Limits and requests for CPU resources are measured in cpu units. In Kubernetes, 1 CPU unit is equivalent to 1 physical CPU core, or 1 virtual core, depending on whether the node is a physical host or a virtual machine running inside a physical machine.

Fractional requests are allowed. When you define a container with spec.containers[].resources.requests.cpu set to 0.5, you are requesting half as much CPU time compared to if you asked for 1.0 CPU. For CPU resource units, the quantity expression 0.1 is equivalent to the expression 100m, which can be read as "one hundred millicpu". Some people say "one hundred millicores", and this is understood to mean the same thing.

CPU resource is always specified as an absolute amount of resource, never as a relative amount. For example, 500m CPU represents the roughly same amount of computing power whether that container runs on a single-core, dual-core, or 48-core machine.

Limits and requests for memory are measured in bytes. You can express memory as a plain integer or as a fixed-point number using one of these quantity suffixes: E, P, T, G, M, k. You can also use the power-of-two equivalents: Ei, Pi, Ti, Gi, Mi, Ki.

The following Pod has two containers. Both containers are defined with a request for 0.25 CPU and 64MiB (226 bytes) of memory. Each container has a limit of 0.5 CPU and 128MiB of memory. You can say the Pod has a request of 0.5 CPU and 128 MiB of memory, and a limit of 1 CPU and 256MiB of memory.



```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: images.my-company.example/app:v4
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
  - name: log-aggregator
    image: images.my-company.example/log-aggregator:v6
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

Figure 3.7
Pod Resource (Kubernetes.io)

When you create a Pod, the Kubernetes scheduler selects a node for the Pod to run on. Each node has a maximum capacity for each of the resource types: the amount of CPU and memory it can provide for Pods. The scheduler ensures that, for each resource type, the sum of the resource requests of the scheduled containers is less than the capacity of the node. Note that although actual memory or CPU resource usage on nodes is very low, the scheduler still refuses to place a Pod on a node if the capacity check fails. This protects against a resource shortage on a node when resource usage later increases, for example, during a daily peak in request rate

When the kubelet starts a container as part of a Pod, the kubelet passes that container's requests and limits for memory and CPU to the container runtime.

On Linux, the container runtime typically configures kernel cgroups that apply and enforce the limits you defined.

The CPU limit defines a hard ceiling on how much CPU time that the container can use. During each scheduling interval (time slice), the Linux kernel checks to see if this limit is exceeded; if so, the kernel waits before allowing that cgroup to resume execution.

The CPU request typically defines a weighting. If several different containers (cgroups) want to run on a contended system, workloads with larger CPU requests are allocated more CPU time than workloads with small requests.

The memory request is mainly used during (Kubernetes) Pod scheduling. On a node that uses cgroups v2, the container runtime might use the memory request as a hint to set memory.min and memory.low.

The memory limit defines a memory limit for that cgroup. If the container tries to allocate more memory than this limit, the Linux kernel out-of-memory subsystem activates and, typically, intervenes by stopping one of the processes in the container that tried to allocate memory. If that process is the container's PID 1, and the container is marked as restartable, Kubernetes restarts the container.

The memory limit for the Pod or container can also apply to pages in memory backed volumes, such as an emptyDir. The kubelet tracks tmpfs emptyDir volumes as container memory use, rather than as local ephemeral storage.

If a container exceeds its memory request and the node that it runs on becomes short of memory overall, it is likely that the Pod the container belongs to will be evicted.

A container might or might not be allowed to exceed its CPU limit for extended periods of time. However, container runtimes don't terminate Pods or containers for excessive CPU usage.

3.5.6 ISTIO

Istio extends Kubernetes to establish a programmable, application-aware network using the powerful Envoy service proxy. Working with both Kubernetes and traditional workloads, Istio brings standard, universal traffic management, telemetry, and security to complex deployments.

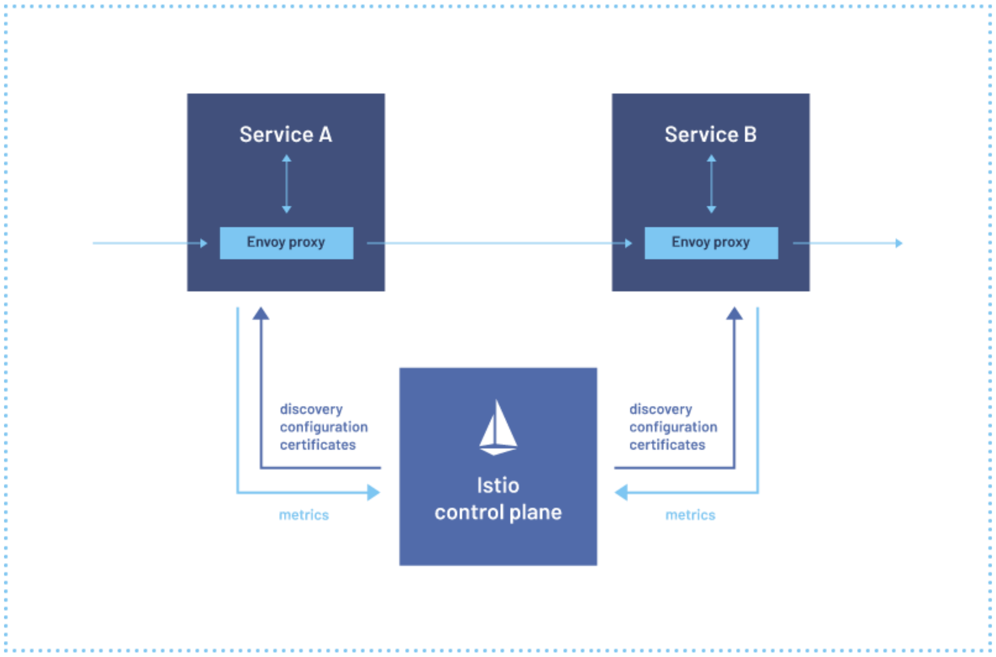


Figure 3.7
Istio Framework (istio.io)

Istio is an open-source service mesh that layers transparently onto existing distributed applications. Istio's powerful features provide a uniform and more efficient way to secure, connect, and monitor services. Istio is the path to load balancing, service-to-service authentication, and monitoring – with few or no service code changes. Its powerful control plane brings vital features, including:

- Secure service-to-service communication in a cluster with TLS encryption, strong identity-based authentication, and authorization
- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic
- Fine-grained control of traffic behaviour with rich routing rules, retries, failovers, and fault injection
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress

3.5.2.1 Traffic management

Routing traffic, both within a single cluster and across clusters, affects performance and enables better deployment strategy. Istio’s traffic routing rules let you easily control the flow of traffic and API calls between services. Istio simplifies configuration of service-level properties like circuit breakers, timeouts, and retries, and makes it easy to set up important tasks like A/B testing, canary deployments, and staged rollouts with percentage-based traffic splits.

The Traffic management as per Istio (istio.io) framework, routing rules let you easily control the flow of traffic and API calls between services. Istio simplifies configuration of service-level properties like circuit breakers, timeouts, and retries, and makes it easy to set up important tasks like A/B testing, canary rollouts, and staged rollouts with percentage-based traffic splits. It also provides out-of-box reliability features that help make your application more resilient against failures of dependent services or the network.

Istio’s traffic management model relies on the Envoy proxies that are deployed along with your services. All traffic that your mesh services send and receive (data plane traffic) is proxied through Envoy, making it easy to direct and control traffic around your mesh without making any changes to your services.

In order to direct traffic within your mesh, Istio needs to know where all your endpoints are, and which services they belong to. To populate its own service registry, Istio connects to a service discovery system. For example, if you’ve installed Istio on a Kubernetes cluster, then Istio automatically detects the services and endpoints in that cluster.

Using this service registry, the Envoy proxies can then direct traffic to the relevant services. Most microservice-based applications have multiple instances of each service workload to handle service traffic, sometimes referred to as a load balancing pool. By default, the Envoy proxies distribute traffic across each service’s load balancing pool using a least requests model, where each request is routed to the host with fewer active requests from a random selection of two hosts from the pool; in this way the most heavily loaded host will not receive requests until it is no more loaded than any other host.

While Istio’s basic service discovery and load balancing gives you a working service mesh, it’s far from all that Istio can do. In many cases you might want more fine-grained control over what happens to your mesh traffic. You might want to direct a particular percentage of traffic to a new version of a service as part of A/B testing or apply a different load balancing policy to traffic for a particular subset of service instances. You might also want to apply special rules to traffic coming into or out of your mesh or add an external dependency of your mesh to the service registry. You can do all this and more by adding your own traffic configuration to Istio using Istio’s traffic management API.

Virtual services, along with destination rules, are the key building blocks of Istio’s traffic routing functionality. A virtual service lets you configure how requests are routed to a service within an Istio service mesh, building on the basic connectivity and discovery provided by Istio and your platform. Each virtual service consists of a set of routing rules that are evaluated in order, letting Istio match each given request to the virtual service to a specific real destination within the mesh. Your mesh can require multiple virtual services or none depending on your use case.

Virtual services play a key role in making Istio’s traffic management flexible and powerful. They do this by strongly decoupling where clients send their requests from the destination workloads that actually implement them. Virtual services also provide a rich way of specifying different traffic routing rules for sending traffic to those workloads.

Why is this so useful? Without virtual services, Envoy distributes traffic using least requests load balancing between all service instances, as described in the introduction. You can improve this behavior with what you know about the workloads. For example, some might represent a different version. This can be useful in A/B testing, where you might want to configure traffic routes based on percentages across different service versions, or to direct traffic from your internal users to a particular set of instances.

With a virtual service, you can specify traffic behavior for one or more hostnames. You use routing rules in the virtual service that tell Envoy how to send the virtual service's traffic to appropriate destinations. Route destinations can be different versions of the same service or entirely different services.

A typical use case is to send traffic to different versions of a service, specified as service subsets. Clients send requests to the virtual service host as if it was a single entity, and Envoy then routes the traffic to the different versions depending on the virtual service rules: for example, "20% of calls go to the new version" or "calls from these users go to version 2". This allows you to, for instance, create a canary rollout where you gradually increase the percentage of traffic that's sent to a new service version. The traffic routing is completely separate from the instance deployment, meaning that the number of instances implementing the new service version can scale up and down based on traffic load without referring to traffic routing at all. By contrast, container orchestration platforms like Kubernetes only support traffic distribution based on instance scaling, which quickly becomes complex. You can read more about how virtual services help with canary deployments in Canary Deployments using Istio.

Virtual services also let you:

- Address multiple application services through a single virtual service. If your mesh uses Kubernetes, for example, you can configure a virtual service to handle all services in a specific namespace. Mapping a single virtual service to multiple “real” services is particularly useful in facilitating turning a monolithic application into a composite service built out of distinct microservices without requiring the consumers of the service to adapt to the transition. Your routing rules can specify “calls to these URIs of monolith.com go to microservice A”, and so on. You can see how this works in one of our examples below.
- Configure traffic rules in combination with gateways to control ingress and egress traffic.

Along with virtual services, destination rules are a key part of Istio’s traffic routing functionality. You can think of virtual services as how you route your traffic to a given destination, and then you use destination rules to configure what happens to traffic for that destination. Destination rules are applied after virtual service routing rules are evaluated, so they apply to the traffic’s “real” destination.

In particular, you use destination rules to specify named service subsets, such as grouping all a given service's instances by version. You can then use these service subsets in the routing rules of virtual services to control the traffic to different instances of your services.

Destination rules also let you customize Envoy's traffic policies when calling the entire destination service or a particular service subset, such as your preferred load balancing model, TLS security mode, or circuit breaker settings. You can see a complete list of destination rule options in the Destination Rule reference.

By default, Istio uses a least requests load balancing policy, where requests are distributed among the instances with the least number of requests. Istio also supports the following models, which you can specify in destination rules for requests to a particular service or service subset.

- Random: Requests are forwarded at random to instances in the pool.
- Weighted: Requests are forwarded to instances in the pool according to a specific percentage.
- Round robin: Requests are forwarded to each instance in sequence.

One can use a gateway to manage inbound and outbound traffic for your mesh, letting you specify which traffic you want to enter or leave the mesh. Gateway configurations are applied to standalone Envoy proxies that are running at the edge of the mesh, rather than sidecar Envoy proxies running alongside your service workloads.

Unlike other mechanisms for controlling traffic entering your systems, such as the Kubernetes Ingress APIs, Istio gateways let you use the full power and flexibility of Istio’s traffic routing. You can do this because Istio’s Gateway resource just lets you configure layer 4-6 load balancing properties such as ports to expose, TLS settings, and so on. Then instead of adding application-layer traffic routing (L7) to the same API resource, you bind a regular Istio virtual service to the gateway. This lets you basically manage gateway traffic like any other data plane traffic in an Istio mesh.

Gateways are primarily used to manage ingress traffic, but you can also configure egress gateways. An egress gateway lets you configure a dedicated exit node for the traffic leaving the mesh, letting you limit which services can or should access external networks, or to enable secure control of egress traffic to add security to your mesh, for example. You can also use a gateway to configure a purely internal proxy.

Istio provides some preconfigured gateway proxy deployments (istio-ingressgateway and istio-egressgateway) that you can use - both are deployed if you use our demo installation, while just the ingress gateway is deployed with our default profile. You can apply your own gateway configurations to these deployments or deploy and configure your own gateway proxies.

One can use a service entry to add an entry to the service registry that Istio maintains internally. After you add the service entry, the Envoy proxies can send traffic to the service as if it was a service in your mesh. Configuring service entries allows you to manage traffic for services running outside of the mesh, including the following tasks:

Redirect and forward traffic for external destinations, such as APIs consumed from the web, or traffic to services in legacy infrastructure.

Define retry, timeout, and fault injection policies for external destinations. Run a mesh service in a Virtual Machine (VM) by adding VMs to your mesh.

You don’t need to add a service entry for every external service that you want your mesh services to use. By default, Istio configures the Envoy proxies to passthrough requests to unknown services.

By default, Istio configures every Envoy proxy to accept traffic on all the ports of its associated workload, and to reach every workload in the mesh when forwarding traffic. You can use a sidecar configuration to do the following:

Fine-tune the set of ports and protocols that an Envoy proxy accepts. Limit the set of services that the Envoy proxy can reach. You might want to limit sidecar reachability like this in larger applications, where having every proxy configured to reach every other service in the mesh can potentially affect mesh performance due to high memory usage.

You can specify that you want a sidecar configuration to apply to all workloads in a particular namespace, or choose specific workloads using a workloadSelector.

A timeout is the amount of time that an Envoy proxy should wait for replies from a given service, ensuring that services don’t hang around waiting for replies indefinitely and that calls succeed or fail within a predictable timeframe. The Envoy timeout for HTTP requests is disabled in Istio by default.

For some applications and services, Istio’s default timeout might not be appropriate. For example, a timeout that is too long could result in excessive latency from waiting for replies from failing services, while a timeout that is too short could result in calls failing unnecessarily while waiting for an operation involving multiple services to return. To find and use your optimal timeout settings, Istio lets you easily adjust timeouts dynamically on a per-service basis using virtual services without having to edit your service code. Here’s a virtual service that specifies a 10 second timeout for calls to the v1 subset of the ratings service:

A retry setting specifies the maximum number of times an Envoy proxy attempts to connect to a service if the initial call fails. Retries can enhance service availability and application performance by making sure that calls don't fail permanently because of transient problems such as a temporarily overloaded service or network. The interval between retries (25ms+) is variable and determined automatically by Istio, preventing the called service from being overwhelmed with requests. The default retry behavior for HTTP requests is to retry twice before returning the error.

Like timeouts, Istio's default retry behavior might not suit your application needs in terms of latency (too many retries to a failed service can slow things down) or availability. Also like timeouts, you can adjust your retry settings on a per-service basis in virtual services without having to touch your service code. You can also further refine your retry behavior by adding per-retry timeouts, specifying the amount of time you want to wait for each retry attempt to successfully connect to the service. The following example configures a maximum of 3 retries to connect to this service subset after an initial call failure, each with a 2 second timeout.

Circuit breakers are another useful mechanism Istio provides for creating resilient microservice-based applications. In a circuit breaker, you set limits for calls to individual hosts within a service, such as the number of concurrent connections or how many times calls to this host have failed. Once that limit has been reached the circuit breaker "trips" and stops further connections to that host. Using a circuit breaker pattern enables fast failure rather than clients trying to connect to an overloaded or failing host.

As circuit breaking applies to “real” mesh destinations in a load balancing pool, you configure circuit breaker thresholds in destination rules, with the settings applying to each individual host in the service. The following example limits the number of concurrent connections for the reviews service workloads of the v1 subset to 100

3.5.2.2 Observability

As services grow in complexity, it becomes challenging to understand behaviour and performance. Istio generates detailed telemetry for all communications within a service mesh. This telemetry provides observability of service behaviour, empowering operators to troubleshoot, maintain, and optimize their applications. Even better, you get almost all this instrumentation without requiring application changes. Through Istio, operators gain a thorough understanding of how monitored services are interacting.

Istio’s telemetry includes detailed metrics, distributed traces, and full access logs. With Istio, you get thorough and comprehensive service mesh observability.

Istio generates detailed telemetry for all service communications within a mesh. This telemetry provides observability of service behaviour, empowering operators to troubleshoot, maintain, and optimize their applications – without imposing any additional burdens on service developers. Through Istio, operators gain a thorough understanding of how monitored services are interacting, both with other services and with the Istio components themselves.

Istio generates the following types of telemetry to provide overall service mesh observability:

- Metrics. Istio generates a set of service metrics based on the four “golden signals” of monitoring (latency, traffic, errors, and saturation). Istio also provides detailed metrics for the mesh control plane. A default set of mesh monitoring dashboards built on top of these metrics is also provided.
- Distributed Traces. Istio generates distributed trace spans for each service, providing operators with a detailed understanding of call flows and service dependencies within a mesh.
- Access Logs. As traffic flows into a service within a mesh, Istio can generate a full record of each request, including source and destination metadata. This information enables operators to audit service behavior down to the individual workload instance level.

Metrics provide a way of monitoring and understanding behavior in aggregate.

To monitor service behavior, Istio generates metrics for all service traffic in, out, and within an Istio service mesh. These metrics provide information on behaviors such as the overall volume of traffic, the error rates within the traffic, and the response times for requests.

In addition to monitoring the behavior of services within a mesh, it is also important to monitor the behavior of the mesh itself. Istio components export metrics on their own internal behaviors to provide insight on the health and function of the mesh control plane.

Istio metrics collection begins with the sidecar proxies (Envoy). Each proxy generates a rich set of metrics about all traffic passing through the proxy (both inbound and outbound). The proxies also provide detailed statistics about the administrative functions of the proxy itself, including configuration and health information.

Envoy-generated metrics provide monitoring of the mesh at the granularity of Envoy resources (such as listeners and clusters). As a result, understanding the connection between mesh services and Envoy resources is required for monitoring the Envoy metrics.

Istio enables operators to select which of the Envoy metrics are generated and collected at each workload instance. By default, Istio enables only a small subset of the Envoy-generated statistics to avoid overwhelming metrics backends and to reduce the CPU overhead associated with metrics collection. However, operators can easily expand the set of collected proxy metrics when required. This enables targeted debugging of networking behavior, while reducing the overall cost of monitoring across the mesh.

Distributed tracing provides a way to monitor and understand behavior by monitoring individual requests as they flow through a mesh. Traces empower mesh operators to understand service dependencies and the sources of latency within their service mesh.

Istio supports distributed tracing through the Envoy proxies. The proxies automatically generate trace spans on behalf of the applications they proxy, requiring only that the applications forward the appropriate request context.

Istio supports a number of tracing backends, including Zipkin, Jaeger, Lightstep, and Datadog. Operators control the sampling rate for trace generation (that is, the rate at which tracing data is generated per request). This allows operators to control the amount and rate of tracing data being produced for their mesh.

3.5.2.3 Security capabilities

Microservices have particular security needs, including protection against man-in-the-middle attacks, flexible access controls, auditing tools, and mutual TLS. Istio includes a comprehensive security solution to give operators the ability to address all these issues. It provides strong identity, powerful policy, transparent TLS encryption, and authentication, authorization, and audit (AAA) tools to protect your services and data.

Istio's security model is based on security-by-default, aiming to provide in-depth defence to allow you to deploy security-minded applications even across distrusted networks.

- To defend against man-in-the-middle attacks, they need traffic encryption.
- To provide flexible service access control, they need mutual TLS and fine-grained access policies.
- To determine who did what at what time, they need auditing tools.

Istio Security provides a comprehensive security solution to solve these issues. This page gives an overview on how you can use Istio security features to secure your services, wherever you run them. In particular, Istio security mitigates both insider and external threats against your data, endpoints, communication, and platform.

The Istio security features provide strong identity, powerful policy, transparent TLS encryption, and authentication, authorization and audit (AAA) tools to protect your services and data. The goals of Istio security are:

- Security by default: no changes needed to application code and infrastructure.
- Defense in depth: integrate with existing security systems to provide multiple layers of defense.
- Zero-trust network: build security solutions on distrusted networks.

Security in Istio involves multiple components:

- A Certificate Authority (CA) for key and certificate management
- The configuration API server distributes to the proxies:
    - authentication policies
    - authorization policies
    - secure naming information
- Sidecar and perimeter proxies work as Policy Enforcement Points (PEPs) to secure communication between clients and servers.
- A set of Envoy proxy extensions to manage telemetry and auditing

Identity is a fundamental concept of any security infrastructure. At the beginning of a workload-to-workload communication, the two parties must exchange credentials with their identity information for mutual authentication purposes. On the client side, the server's identity is checked against the secure naming information to see if it is an authorized runner of the workload. On the server side, the server can determine what information the client can access based on the authorization policies, audit who accessed what at what time, charge clients based on the workloads they used, and reject any clients who failed to pay their bill from accessing the workloads.

The Istio identity model uses the first-class service identity to determine the identity of a request's origin. This model allows for great flexibility and granularity for service identities to represent a human user, an individual workload, or a group of workloads. On platforms without a service identity, Istio can use other identities that can group workload instances, such as service names.

The following list shows examples of service identities that you can use on different platforms:

- Kubernetes: Kubernetes service account
- GCE: GCP service account
- On-premises (non-Kubernetes): user account, custom service account, service name, Istio service account, or GCP service account. The custom service account refers to the existing service account just like the identities that the customer's Identity Directory manages.

Istio provides two types of authentication:

- Peer authentication: used for service-to-service authentication to verify the client making the connection. Istio offers mutual TLS as a full stack solution for transport authentication, which can be enabled without requiring service code changes. This solution:
    - Provides each service with a strong identity representing its role to enable interoperability across clusters and clouds.
    - Secures service-to-service communication.
    - Provides a key management system to automate key and certificate generation, distribution, and rotation.
- Request authentication: Used for end-user authentication to verify the credential attached to the request. Istio enables request-level authentication with JSON Web

Token (JWT) validation and a streamlined developer experience using a custom authentication provider or any OpenID Connect providers, for example:

- ORY Hydra
- Keycloak
- Auth0
- Firebase Auth
- Google Auth

In all cases, Istio stores the authentication policies in the Istio config store via a custom Kubernetes API. Istiod keeps them up to date for each proxy, along with the keys where appropriate. Additionally, Istio supports authentication in permissive mode to help you understand how a policy change can affect your security posture before it is enforced.

Istio mutual TLS has a permissive mode, which allows a service to accept both plaintext traffic and mutual TLS traffic at the same time. This feature greatly improves the mutual TLS onboarding experience.

Many non-Istio clients communicating with a non-Istio server presents a problem for an operator who wants to migrate that server to Istio with mutual TLS enabled. Commonly, the operator cannot install an Istio sidecar for all clients at the same time or does not even have the permissions to do so on some clients. Even after installing the Istio sidecar on the server, the operator cannot enable mutual TLS without breaking existing communications.

With the permissive mode enabled, the server accepts both plaintext and mutual TLS traffic. The mode provides greater flexibility for the on-boarding process. The server’s installed Istio sidecar takes mutual TLS traffic immediately without breaking existing plaintext traffic. As a result, the operator can gradually install and configure the client’s Istio sidecars to send mutual TLS traffic. Once the configuration of the clients is complete, the operator can configure the server to mutual TLS only mode.

You can specify authentication requirements for workloads receiving requests in an Istio mesh using peer and request authentication policies. The mesh operator uses .yaml files to specify the policies. The policies are saved in the Istio configuration storage once deployed. The Istio controller watches the configuration storage.

Upon any policy changes, the new policy is translated to the appropriate configuration telling the PEP how to perform the required authentication mechanisms. The control plane may fetch the public key and attach it to the configuration for JWT validation. Alternatively, Istiod provides the path to the keys and certificates the Istio system manages and installs them to the application pod for mutual TLS. You can find more info in the Identity and certificate management section.

Istio sends configurations to the targeted endpoints asynchronously. Once the proxy receives the configuration, the new authentication requirement takes effect immediately on that pod.

Client services, those that send requests, are responsible for following the necessary authentication mechanism. For request authentication, the application is responsible for acquiring and attaching the JWT credential to the request. For peer authentication, Istio automatically upgrades all traffic between two PEPs to mutual TLS. If authentication policies disable mutual TLS mode, Istio continues to use plain text between PEPs. To override this behavior explicitly disable mutual TLS mode with destination rules.

Istio’s authorization features provide mesh-, namespace-, and workload-wide access control for your workloads in the mesh. This level of control provides the following benefits:

- Workload-to-workload and end-user-to-workload authorization.
- A simple API: it includes a single AuthorizationPolicy CRD, which is easy to use and maintain.
- Flexible semantics: operators can define custom conditions on Istio attributes, and use CUSTOM, DENY and ALLOW actions.

- High performance: Istio authorization (ALLOW and DENY) is enforced natively on Envoy.
- High compatibility: supports gRPC, HTTP, HTTPS and HTTP/2 natively, as well as any plain TCP protocols.

3.6 Data Collection Procedures

Data for this research was collected from two sources, namely, primary, and secondary. Primary data is typically defined as the first occurrence of a piece of work. For this research, primary data is obtained from the open-source tools such as Intel V-Tune Profiler. The configuration of the tools for Hot Spot analysis makes the data to be narrowed down and quite suitable for further analysis.

Secondary data is obtained from literature review, where the researcher has perused several articles, journals, papers, and books to gain as much information as possible that is relevant to this research.

The researcher thoroughly investigated the tools used for data generation. The configuration of the tools for cryptocurrency mining process was focus.

To analyse the data collected, the researcher prepared pipeline from source of data generation, data cleaning to get the suitable data format. Working on a framework to store and retrieve data, building a logic to search the frequency of certain key words, reducing redundancies, and removing irrelevant data. By using these procedures, the researcher was able to reduce the data collected to only relevant data to be analysed. More info about the data collection and analysis is described in further section in detail.

3.7 Data Analysis

Data collection for quantitative research very often involve data in the form of counts or numbers where each data set has a unique numerical value. This data is any quantifiable information that researchers can use for mathematical calculations and statistical analysis to make real-life decisions based on these mathematical derivations. This data however must be analyzed to make sense of. There are multiple methods of analyzing quantitative data collected as follows (Quantitative data):

- **Cross-tabulation**: It is the most widely used quantitative data analysis methods. It is a preferred method since it uses a basic tabular form to draw inferences between different data-sets in the research study. It contains data that is mutually exclusive or have some connection with each other.
- **Trend analysis**: Trend analysis is a statistical analysis method that provides the ability to look at quantitative data that has been collected over a long period of time. This data analysis method helps collect feedback about data changes over time and if aims to understand the change in variables considering one variable remains unchanged.
- **MaxDiff analysis:** The MaxDiff analysis is a quantitative data analysis method that is used to gauge customer preferences for a purchase and what parameters rank higher than the others in this process. In a simplistic form, this method is also called the “best-worst” method. This method is very similar to conjoint analysis but is much easier to implement and can be interchangeably used.
- **Conjoint analysis:** Like in the above method, conjoint analysis is a similar

quantitative data analysis method that analyses parameters behind a purchasing decision. This method possesses the ability to collect and analyse advanced metrics which provide an in-depth insight into purchasing decisions as well as the parameters that rank the most important.

- **TURF analysis:** TURF analysis or Total Unduplicated Reach and Frequency Analysis, is a quantitative data analysis methodology that assesses the total market reach of a product or service or a mix of both. This method is used by organizations to understand the frequency and the avenues at which their messaging reaches customers and prospective customers which helps them tweak their go-to-market strategies.
- **Gap analysis:** Gap analysis uses a side-by-side matrix to depict data that helps measure the difference between expected performance and actual performance. This data analysis helps measure gaps in performance and the things that are required to be done to bridge this gap.
- **SWOT analysis:** SWOT analysis, is a quantitative data analysis method that assigns numerical values to indicate strength, weaknesses, opportunities and threats of an organization or product or service which in turn provides a holistic picture about competition. This method helps to create effective business strategies.
- **Text analysis:** Text analysis is an advanced statistical method where intelligent tools make sense of and quantify or fashion qualitative and open-ended data into easily understandable data. This method is used when the raw survey data is unstructured but must be brought into a structure that makes sense.

Steps to conduct Quantitative Data Analysis:

- For Quantitative Data, raw information must be presented in a meaningful manner using data analysis methods. This data should be analysed to find evidential data that would help in the research process.
- Associate measurement scales such as Nominal, Ordinal, Interval and Ratio with the variables. This step is important to arrange the data in proper order. Data can be entered into an excel sheet to organize it in a specific format.
- Link descriptive statistics to encapsulate available data. It can be difficult to establish a pattern in the raw data. Some widely used descriptive statistics are:
  - Mean- An average of values for a specific variable
  - Median- A midpoint of the value scale for a variable
  - Mode- For a variable, the most common value
  - Frequency- Number of times a particular value is observed in the scale
  - Minimum and Maximum Values- Lowest and highest values for a scale
  - Percentages- Format to express scores and set of values for variables

It is important to decide the measurement scale to conclude descriptive statistics for the variable. For instance, a nominal variable score will never have a mean or median and so the descriptive statistics will correspondingly vary. Descriptive statistics suffice in situations where the results are not to be generalized to the population.

Select appropriate tables to represent data and analyze collected data: After deciding on a suitable measurement scale, researchers can use a tabular format to represent data. This data can be analyzed using various techniques such as Cross-tabulation or TURF.

In summary, the author of this research will follow the above steps, especially categorization of the data. To accomplish this, the author has adopted the Text Analysis to ensure the correct prediction for the crypto miner process. Multiple analyses are also required so that the context of each output is not missed or misinterpreted by the author, highlighting key output and information that may have been overlooked in previous readings.

3.8 Coding and analysis

The researcher derived a coding process based on the work of Taylor-Powell (2003). The coding process used in this research was divided into the following phases

- **The review phase** – The researcher reviewed the logs of the customer application monitored by the Intel V-Tune profiler several times to get familiarised with the data. During the review phase, several fine tunings to the tool's configuration has done accordingly.
- **The coding phase** – The output of the Intel V-Tune has clearly will be in the raw format and the prepossessing the log to CSV format is very much necessary and important phase of the research. Once the log data is made a proper table format will be ready for consumption by the parser to decide the results.
- **The analysis phase** – Here the researcher uses the CSV Format file for further parsing logic that decides the application is affected by the mining logic or not. Based on the output, if the application is affected, then the process continues in the pipeline for Istio framework. The researcher uses the logic and text analysis method to derive the deciding factor.

3.9 Methods of validation

Following environment are considered while evaluating the data.

- Hyperledger Framework is chosen as the case study for Blockchain Deployment.
- The basic software images or binaries will be of Docker images.
- Istio Framework is employed to mitigate the risk of zero downtime in production environment.
- XMRIG Client is used to simulate the miner process.
- Intel V-Tune Profile is used to inspect the miner process.
- Custom Decider Logic is employed to check the logs/reports and conclude the results.
- Repeat the steps from top if necessary for other process.

The quality of a research depends on whether the research can withstand the test of reliability and validity. According to the work of Shoaib and Mujtaba in 2016, the research must address the components of dependability, transferability, credibility, and conformability (Shoaib, Mujtaba, 2016).

In a quantitative analysis, the researcher achieves reliability and validity by ensuring that the data is genuine. The researcher analyzed each log of the mining process and concluded the stack trace that always remains the same. Reliability of a research can be defined as the likelihood of the results of similar research, using the same parameters would yield the same results. By using multiple data collection methods, the risk of a single approach is minimized.

To ensure that the results of the research are trustworthy, the researcher must adhere to the principles of integrity, transferability, and reliability. According to Robert K. Yin , (2015) The validity of a research is dependent on the quality of the research process, and therefore must not be impaired at any stage. Therefore, the validity of the research is a metric to measure the quality of the research.

Because quantitative analyses depend on the numeric data, the results and findings of the researcher must be consistent and verifiable if they are to be contribute to existing knowledge and create additional proposals for research in the future.

3.10 Research Design Limitations

This research does have some limitations that may reduce the generalization of the findings. Since this was a study conducted on the x86 based hardware and assumptions were made that the same behavior would result on other hardware platforms also.

Furthermore, the Intel V-Tune assumes that the detecting the mining process in the test bed would be done mostly on X-86 architectures. The tool Intel V-Tune will not work other than X-86 architectures. So, it is advised to make sure that, before the tool is used for testing purpose, a user manual and compatibly check is carried out of the tool and supported platform.

3.11 Conclusion

The research findings have explored quantitative method of research design. The research instruments used for this research were readily available open-source tools. Also, the research has taken care of supporting any x86 platform architecture to support. End to End pipeline has been discussed with clear explanations of each framework components. Finally, the limitations of the research were stated and discussed.

CHAPTER IV:
RESULTS AND FINDINGS

4.1 Introduction

This chapter covers the results of the research and its major findings. This chapter is roughly divided into two parts: the first part covers the research case, where details about the Intel V-Tune configurations, Xmrig (Cryptomining application) parameters, other CPU hog applications run. The second part consists of the data analysis performed by the researcher after the logs of the applications monitored by the Intel V-Tune profiler were collected.

4.2 The Research Case

This section details the research insights and the pipeline involved from source of data generation till the analysis is concluded. Following theoretical aspects is chosen for the steps to be taken.

.

4.2.1 Cryptomining

Crypto mining (Crypto mining) simply is not only as a way of creating new digital currency. Crypto mining, however, also involves validating cryptocurrency transactions on a blockchain network and adding them to a distributed ledger. Most importantly, crypto mining prevents the double-spending of digital currency on a distributed network.

Like physical currencies, when one member spends cryptocurrency, the digital ledger must be updated by debiting one account and crediting the other. However, the challenge of a digital currency is that digital platforms are easily manipulated. Bitcoin’s distributed ledger, therefore, only allows verified miners to update transactions on the digital ledger. This gives miners the extra responsibility of securing the network from double spending.

Meanwhile, new coins are generated to reward miners for their work in securing the network. Since distributed ledgers lack a centralized authority, the mining process is crucial for validating transactions. Miners are, therefore, incentivized to secure the network by participating in the transaction validation process that increases their chances of winning newly minted coins.

To ensure that only verified crypto miners can mine and validate transactions, a proof-of-work (PoW) consensus protocol has been put into place. PoW also secures the network from any external attacks.

4.2.2 PROOF-OF-WORK

Mining the coin will trigger the release of new coins into circulation. For miners to be rewarded with new coins, they need to deploy machines that solve complex mathematical equations in the form of cryptographic hashes. A hash is a truncated digital signature of a chunk of data. Hashes are generated to secure data transferred on a public network. Miners compete with their peers to zero in on a hash value generated by a crypto coin transaction, and the first miner to crack the code gets to add the block to the ledger and receive the reward.

Each block uses a hash function to refer to the previous block, forming an unbroken chain of blocks that leads back to the first block. For this reason, peers on the network can easily verify whether certain blocks are valid and whether the miners who validated each block properly solved the hash to receive the reward.

Over time, as miners deploy more advanced machines to solve PoW, the difficulty of equations on the network increases. At the same time, competition among miners rises, increasing the scarcity of the cryptocurrency as a result.

4.2.3 CPU Based CryptoMining

Cryptocurrencies mining requires computers with special software specifically designed to solve complicated, cryptographic mathematic equations. In the technology’s early days, cryptocurrencies like Monero could be mined with a simple CPU chip on a home computer. Over the years, however, CPU chips have become impractical for mining most cryptocurrencies due to the increasing difficulty levels.

Today, mining cryptocurrencies requires a specialized GPU or an application-specific integrated circuit (ASIC) miner. In addition, the GPUs in the mining rig must be always connected to a reliable internet connection. Each crypto miner is also required to be a member of an online crypto mining pool as well.

CPU mining was the go-to option for most miners. As it is easily available hardware for mining. Any decent hardware x86 based configuration were used in early days for crypto mining.

GPU mining is another method of mining cryptocurrency. It maximizes computational power by bringing together a set of GPUs under one mining rig. For GPU mining, a motherboard and cooling system is required for the rig.

Similarly, ASIC mining is yet another method of mining cryptocurrencies. Unlike GPU miners, ASIC miners are specifically designed to mine cryptocurrencies, so they produce more cryptocurrency units than GPUs. However, they are expensive, meaning that, as mining difficulty increases, they quickly become obsolete.

Given the ever-increasing costs of GPU and ASIC mining, cloud mining is becoming increasingly popular. Cloud mining allows individual miners to leverage the power of major corporations and dedicated crypto mining facilities.

Pooling the hardware resources and mining the block will be beneficial as it reduces the need of capacity of hardware resources. The rewards for mining will be distributed among the pool participants in proportion to the number of resources that each miner contributed to the pool. Miners consider official crypto mining pools more reliable since they receive frequent upgrades by their host companies, as well as regular technical support. The best place to find mining pools is CryptoCompare, where miners can compare different mining pools based on their reliability, profitability, and the coin that they want to mine.

Prospective miner chooses a CPU, GPU, ASIC miner, or cloud mining, the most important factors to consider are the mining rig's hash rate, electric power consumption, and overall costs. Generally, crypto mining machines consume a considerable amount of electricity and emit significant heat.

For instance, the average ASIC miner will use about 72 terawatts of power to create a bitcoin in about ten minutes. These figures continue to change as technology advances and mining difficulty increases.

Even though the price of the machine matters, it is just as important to consider electricity consumption, electricity costs in the area, and cooling costs, especially with GPU and ASIC mining rigs.

It is also important to consider the level of difficulty for the cryptocurrency that an individual wants to mine, in order determine whether the operation would even be profitable.

For the research XMRig is used to demonstrate the cryptojacking.
XMRig is a high performance, open source, cross platform RandomX, KawPow, CryptoNight, AstroBWT and GhostRider unified CPU/GPU miner and RandomX benchmark. Official binaries are available for Windows, Linux, macOS and FreeBSD.

- Mining backends
- **CPU** (x64/ARMv7/ARMv8)
- **OpenCL** for AMD GPUs.
- **CUDA** for NVIDIA GPUs via external CUDA plugin.

4.2.4 Business and Social Impacts of CPU based Cryptojacking

Following are the points found as part of the business and social impacts from nacdl.org (1986).

<u>Legal and Regulatory Impact:</u>

- Unauthorized access and computer crimes:
    - The CFAA (Computer Fraud and Abuse Act) prohibits unauthorized access to protected computers and networks. CPU-based cryptojacking, which involves unauthorized use of computing resources, may fall under this law, especially if the perpetrator gains access through hacking or unauthorized means.
    - CPU-based cryptojacking involves unauthorized access to computer systems and the use of computing resources without permission. This activity may be considered a computer crime, and laws related to unauthorized access, hacking, or computer fraud may apply. Identify relevant laws and regulations in your jurisdiction that address these issues.
- Data Protection Laws (General Data Protection Regulation - GDPR):
    - CPU-based cryptojacking can lead to the exposure or unauthorized access of personal data, which may violate data protection laws. Businesses are responsible for protecting personal data and may face penalties if they fail to implement appropriate security measures or adequately respond to incidents.

- Cybersecurity and Data Breach Notification Laws:
    - Many jurisdictions have specific regulations that require businesses to implement reasonable cybersecurity measures and notify individuals in the event of a data breach. Cryptojacking incidents may trigger these requirements if they involve unauthorized access and potential data exposure.
    - If a business detects a cryptojacking incident resulting in the compromise of customer data, they may be obligated to notify affected individuals and regulatory authorities in accordance with data breach notification laws.
- International legal considerations:
    - If businesses operating across international borders, consider the legal implications of cryptojacking in different jurisdictions. Laws and regulations can vary significantly, so it's important to understand the legal landscape in each relevant jurisdiction.
- Intellectual property implications:
    - Cryptojacking malware is often distributed through malicious websites, emails, or compromised software, potentially infringing on intellectual property rights. Assess the legal implications related to the distribution of malicious software and potential copyright or trademark infringement issues.

- Consumer protection and fraud:
    - If cryptojacking affects consumers or customers, consumer protection laws and regulations may come into play. Evaluate regulations related to unfair or deceptive trade practices, fraud, or consumer rights to understand the potential legal implications for businesses.

<u>Economic Impact :</u>

According to aquasec.com (2004), following points can be derived after careful analysis

- System Performance:
    - Cryptojacking activities consume a considerable amount of CPU resources, which can cause noticeable performance degradation of affected systems. Slower processing speeds and increased system instability can lead to decreased productivity and efficiency, particularly for businesses relying on computer-intensive tasks.
- Hardware life degradation:
    - Continuous mining operations using CPUs can place excessive strain on hardware components, potentially shortening their lifespan. As a result, individuals or organizations may need to incur additional expenses for repairing or replacing affected hardware, further impacting their budgets.
- Opportunity costs:
    - CPU-based cryptojacking siphons off computing power and resources that could have been used for legitimate tasks or operations. This opportunity cost translates into lost productivity and potentially missed business opportunities.
- Power consumption:
    - CPU-based cryptojacking utilizes significant computational power, leading to a substantial increase in electricity consumption. This can result in higher energy bills for affected individuals or organizations, impacting their operational costs.

- Choking in Network:
    - In large-scale cryptojacking incidents, where numerous systems are compromised within a network, the increased network traffic can lead to congestion and slower network performance. This can affect communication, data transfer, and overall network efficiency
- Trust Factor:
    - Organizations that fall victim to CPU-based cryptojacking may suffer reputational damage and a loss of trust from their customers or stakeholders. This can impact their brand image, customer loyalty, and long-term business relationships, potentially leading to financial consequences
- Expense towards security measures:
    - Implementing countermeasures and mitigation strategies to detect and prevent CPU-based cryptojacking requires investments in security solutions, software updates, employee training, and infrastructure enhancements. These costs should also be factored into the overall economic impact analysis.

Strategic Implications on various sectors of Business :

According to enterslice.com (2024), digital banking implications because of cryptojacking can be applied in general for following sectors

- Financial Sector:
    - Strategic Implications: In the financial sector, CPU-based cryptojacking can disrupt critical systems, impact transaction processing speed, and compromise data integrity. This can undermine customer trust and confidence in the security of financial services.
    - Strategic Considerations: Financial institutions need to prioritize robust cybersecurity measures to prevent and detect cryptojacking incidents. They should invest in real-time monitoring, intrusion detection systems, and employee training to enhance their defense against such threats.
- Healthcare Sector:
    - Strategic Implications: CPU-based cryptojacking can severely affect healthcare organizations by slowing down critical systems, leading to delays in patient care, and compromising sensitive medical data. It can also disrupt medical research activities and the availability of life-saving services.
    - Strategic Considerations: Healthcare organizations must prioritize cybersecurity measures, including regular system patching, network segmentation, and staff education on identifying and reporting suspicious activities. Protecting patient data and ensuring uninterrupted delivery of healthcare services should be top priorities.

- IT and Technology Sector:
    - Strategic Implications: CPU-based cryptojacking can have significant consequences for IT and technology companies, impacting their own systems and potentially leading to reputational damage if their products or services are exploited by attackers. It can also divert valuable computing resources from core business operations.
    - Strategic Considerations: IT and technology companies need to focus on secure software development practices, regular vulnerability assessments, and prompt security updates to mitigate the risk of their products being used for cryptojacking. Implementing robust intrusion detection and prevention systems is crucial to safeguard their infrastructure and customer data.
- E-commerce and Retail Sector:
    - Strategic Implications: In the e-commerce and retail sector, CPU-based cryptojacking can affect website performance, leading to slow page load times, poor user experience, and potentially impacting sales and customer retention. It can also compromise customer payment information.
    - Strategic Considerations: E-commerce and retail businesses should prioritize website security and implement strong access controls. Regular vulnerability scanning, secure payment processing, and customer communication regarding security measures can help maintain customer trust and protect their sensitive data.

- Energy Sector:
    - Strategic Implications: CPU-based cryptojacking in the energy sector can lead to increased electricity consumption, impacting operational costs and energy efficiency goals. It can also disrupt critical infrastructure and result in power outages or other operational failures.
    - Strategic Considerations: Energy companies should focus on securing their control systems, implementing network segmentation, and investing in intrusion detection and prevention mechanisms. Regular monitoring of energy consumption patterns can help identify abnormal usage indicative of cryptojacking activities.

- Education Sector:
    - Strategic Implications: CPU-based cryptojacking can impact educational institutions by slowing down computer systems, hindering academic activities, and potentially compromising sensitive student and faculty data.
    - Strategic Considerations: Educational institutions should raise awareness among students, faculty, and staff about safe computing practices and the risks of cryptojacking. Robust cybersecurity measures, including system monitoring and strong access controls, should be in place to protect the integrity of academic operations and student information.

Ethics and Social responsibility :

The cointelegraph.com (2018) collects the points as below for ethics of cryptojacking

- Cybercrime and Criminal Profit:
    - CPU-based cryptojacking is often linked to cybercriminal activities. This raises ethical questions regarding the exploitation of technology for personal gain, contributing to a broader landscape of cybercrime.

- Educational Awareness and Digital Literacy:
    - Cryptojacking highlights the need for increased awareness and education about online security, digital threats, and safe computing practices. Enhancing digital literacy can empower individuals and organizations to protect themselves against such risks.

- Regulatory and Legal Considerations:
    - Ethical and social implications may drive the need for regulatory frameworks and legislation to address CPU-based cryptojacking, ensuring appropriate consequences for perpetrators and providing protections for victims.

- Impact on System Performance and User Experience:
    - Cryptojacking can degrade system performance, resulting in slow response times, increased heat generation, and reduced battery life. This negatively affects user experience, productivity, and the usability of devices.

- Public Perception of Cryptocurrencies:
    - Cryptojacking incidents can influence public perception of cryptocurrencies, associating them with illicit activities and unethical behaviour. This perception may impact the broader acceptance and adoption of cryptocurrencies as a legitimate form of digital currency.
- Inequitable Distribution of Costs:
    - Cryptojacking imposes costs, such as increased electricity bills and potential hardware damage, on the victims. The burden falls disproportionately on individuals and organizations that may be less equipped to handle the financial impact.

- Trust and Security:
    - Cryptojacking undermines trust in digital systems and cybersecurity. It raises concerns about the security and integrity of software and websites, impacting individuals' and organizations' confidence in online transactions and interactions.

- Resource Misuse:
    - Cryptojacking exploits the computational resources of individuals and organizations without their permission. This misuse of resources can be seen as unethical, as it diminishes the intended purpose of those resources and can disrupt normal operations.
- Power Consumption and Environmental Impact:
    - CPU-based cryptojacking significantly increases power consumption, leading to increased carbon footprint and energy waste. This has environmental consequences and conflicts with sustainability goals.

4.3 Data Analysis

E2E is the pipeline that explains the end-to-end process for identifying the cryptojacking. It includes the steps for from source of data collection, data cleaning, report generation and final data analysis.

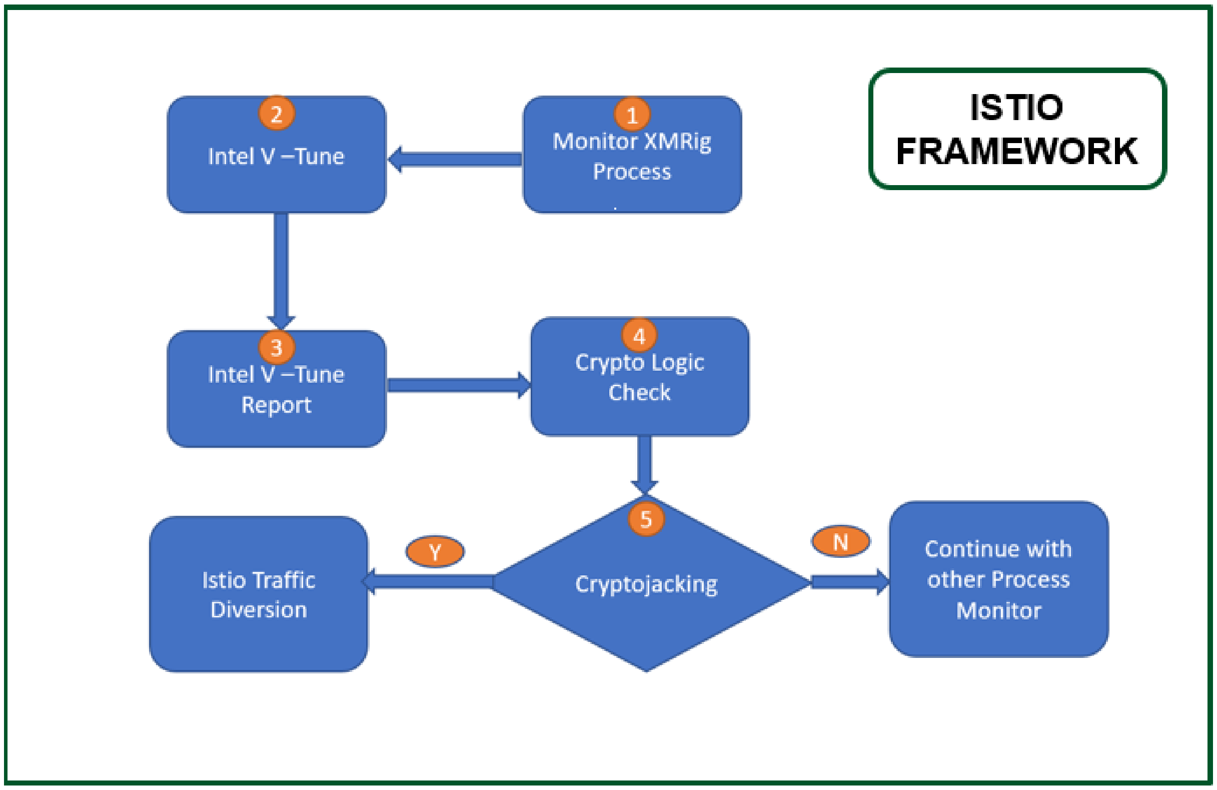


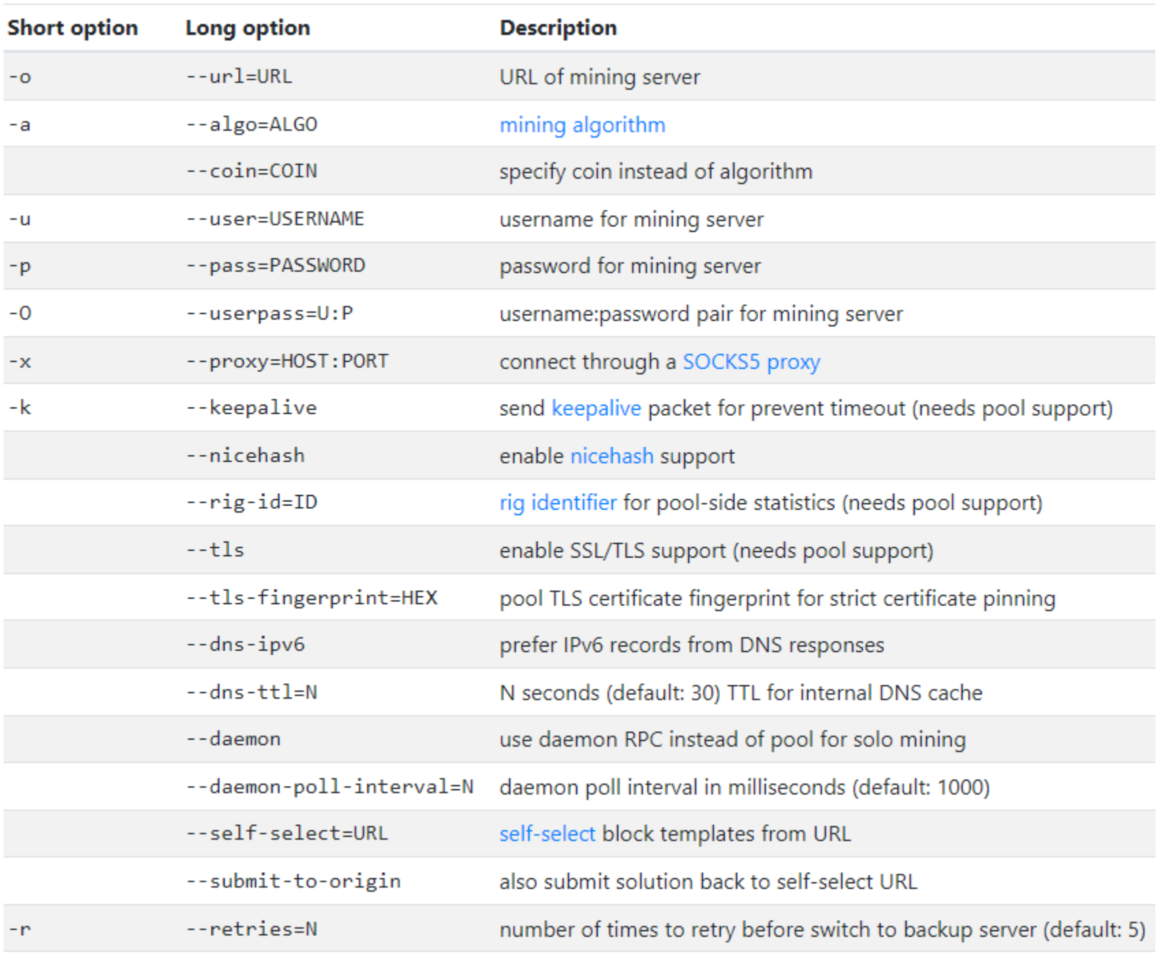
Figure 4.1
E2E Pipeline

Further following points explain the technical part at high level. Detailed explanation will follow in the subsequent sections.

- Start the XMRig process with the shared pool mode (Pools are groups of miners who share their computational resources. Mining pools utilize these combined resources to strengthen the probability of finding a block).
- Configure the Intel V T-Tune to monitor the XMRig process.
- After the desired duration of monitor, CSV based report is generated.
- Logic for checking the Cryptojacking is started.
- If “yes” for Cryptojacking, then the affected traffic is diverted to Istio framework else “No” for further monitor of other process to monitor.

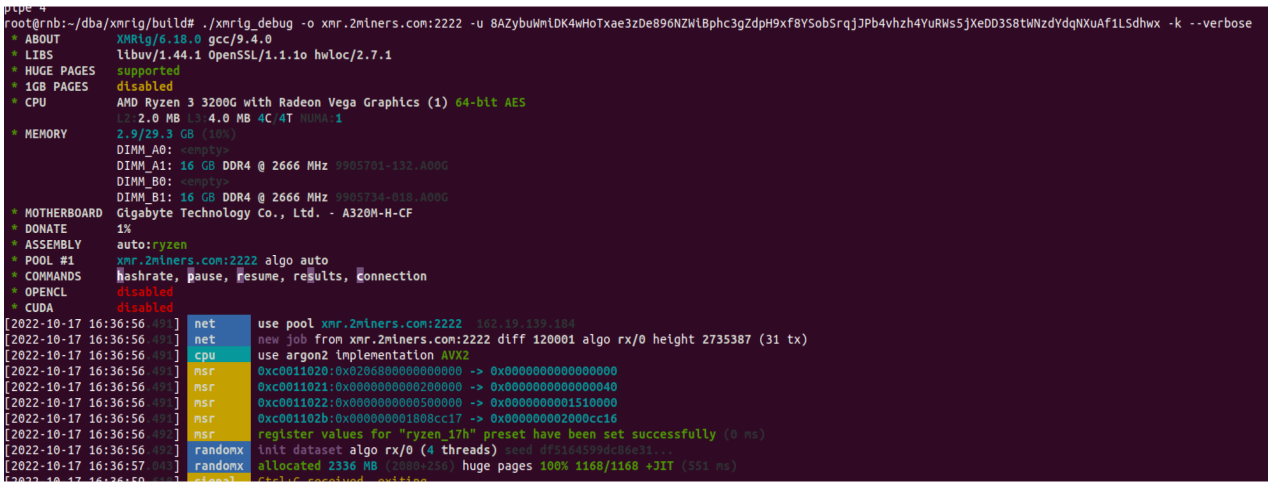
4.4 Pipeline stages

1) Launch the XMRig process with options available as below



Short option	Long option	Description
`-o`	`--url=URL`	URL of mining server
`-a`	`--algo=ALGO`	mining algorithm
	`--coin=COIN`	specify coin instead of algorithm
`-u`	`--user=USERNAME`	username for mining server
`-p`	`--pass=PASSWORD`	password for mining server
`-O`	`--userpass=U:P`	username:password pair for mining server
`-x`	`--proxy=HOST:PORT`	connect through a SOCKS5 proxy
`-k`	`--keepalive`	send keepalive packet for prevent timeout (needs pool support)
	`--nicehash`	enable nicehash support
	`--rig-id=ID`	rig identifier for pool-side statistics (needs pool support)
	`--tls`	enable SSL/TLS support (needs pool support)
	`--tls-fingerprint=HEX`	pool TLS certificate fingerprint for strict certificate pinning
	`--dns-ipv6`	prefer IPv6 records from DNS responses
	`--dns-ttl=N`	N seconds (default: 30) TTL for internal DNS cache
	`--daemon`	use daemon RPC instead of pool for solo mining
	`--daemon-poll-interval=N`	daemon poll interval in milliseconds (default: 1000)
	`--self-select=URL`	self-select block templates from URL
	`--submit-to-origin`	also submit solution back to self-select URL
`-r`	`--retries=N`	number of times to retry before switch to backup server (default: 5)

Figure 4.2
XMRig Process options



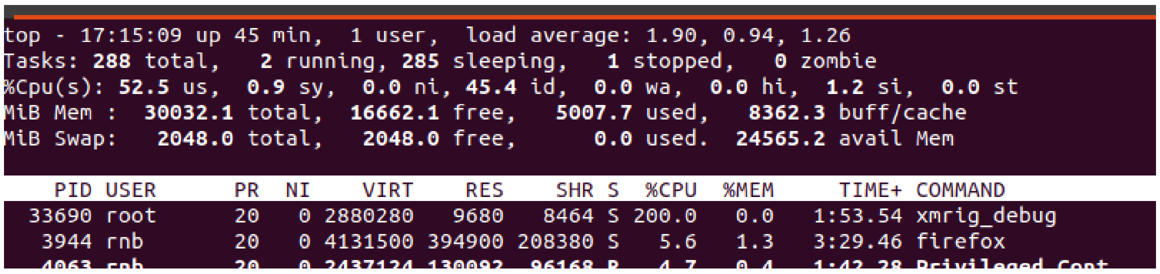
```
pipe 4
root@rnb:~/dba/xmrig/build# ./xmrig_debug -o xmr.2miners.com:2222 -u 8AZybuWmiDK4wHoTxae3zDe896NZWiBphc3gZdpH9xf8YSobSrqjJPb4vhzh4YuRWs5jXeDD3S8tWNzdYdqNXuAf1LSdhwx -k --verbose
 * ABOUT        XMRig/6.18.0 gcc/9.4.0
 * LIBS         libuv/1.44.1 OpenSSL/1.1.1o hwloc/2.7.1
 * HUGE PAGES   supported
 * 1GB PAGES    disabled
 * CPU          AMD Ryzen 3 3200G with Radeon Vega Graphics (1) 64-bit AES
                L2:2.0 MB L3:4.0 MB 4C/4T NUMA:1
 * MEMORY       2.9/29.3 GB (10%)
                DIMM_A0: <empty>
                DIMM_A1: 16 GB DDR4 @ 2666 MHz 9905701-132.A00G
                DIMM_B0: <empty>
                DIMM_B1: 16 GB DDR4 @ 2666 MHz 9905734-018.A00G
 * MOTHERBOARD  Gigabyte Technology Co., Ltd. - A320M-H-CF
 * DONATE       1%
 * ASSEMBLY     auto:ryzen
 * POOL #1      xmr.2miners.com:2222 algo auto
 * COMMANDS     hashrate, pause, resume, results, connection
 * OPENCL       disabled
 * CUDA         disabled
[2022-10-17 16:36:56.491]  net       use pool xmr.2miners.com:2222  162.19.139.184
[2022-10-17 16:36:56.491]  net       new job from xmr.2miners.com:2222 diff 120001 algo rx/0 height 2735387 (31 tx)
[2022-10-17 16:36:56.491]  cpu       use argon2 implementation AVX2
[2022-10-17 16:36:56.491]  msr       0xc0011020:0x0206800000000000 -> 0x0000000000000000
[2022-10-17 16:36:56.491]  msr       0xc0011021:0x0000000000200000 -> 0x0000000000000040
[2022-10-17 16:36:56.491]  msr       0xc0011022:0x0000000000500000 -> 0x0000000001510000
[2022-10-17 16:36:56.491]  msr       0xc001102b:0x000000001808cc17 -> 0x000000002000cc16
[2022-10-17 16:36:56.492]  msr       register values for "ryzen_17h" preset have been set successfully (0 ms)
[2022-10-17 16:36:56.492]  randomx   init dataset algo rx/0 (4 threads) seed df5164599dc86e31...
[2022-10-17 16:36:57.043]  randomx   allocated 2336 MB (2080+256) huge pages 100% 1168/1168 +JIT (551 ms)
[2022-10-17 16:36:59.618]  signal    Ctrl+C received, exiting
```

Figure 4.3
Intel V-Tune Profile – XMRig Launch

2) Measure the CPU consumption of the crypto process

To know what application process to measure, check the most CPU utilizing application using the OS specific commands. For Ex: Linux platform use **"top"** command

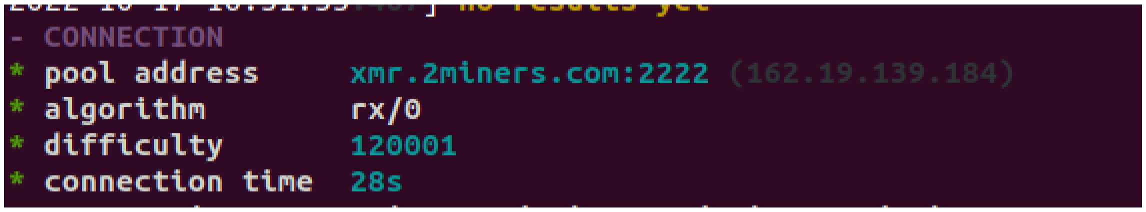
Measure the CPU utilization by the XMRig process using the **"top"** command. In below figure the XMRig process shows 200% CPU utilization which leads to suspect the activity.



```
top - 17:15:09 up 45 min,  1 user,  load average: 1.90, 0.94, 1.26
Tasks: 288 total,   2 running, 285 sleeping,   1 stopped,   0 zombie
%Cpu(s): 52.5 us,  0.9 sy,  0.0 ni, 45.4 id,  0.0 wa,  0.0 hi,  1.2 si,  0.0 st
MiB Mem :  30032.1 total,  16662.1 free,   5007.7 used,   8362.3 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.  24565.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  33690 root      20   0 2880280   9680   8464 S 200.0   0.0   1:53.54 xmrig_debug
   3944 rnb       20   0 4131500 394900 208380 S   5.6   1.3   3:29.46 firefox
   4063 rnb       20   0 2437124 130092  96168 R   4.7   0.4   1:42.28 Privileged Cont
```

Figure 4.4
Intel V-Tune Profile – XMRig CPU Utilization

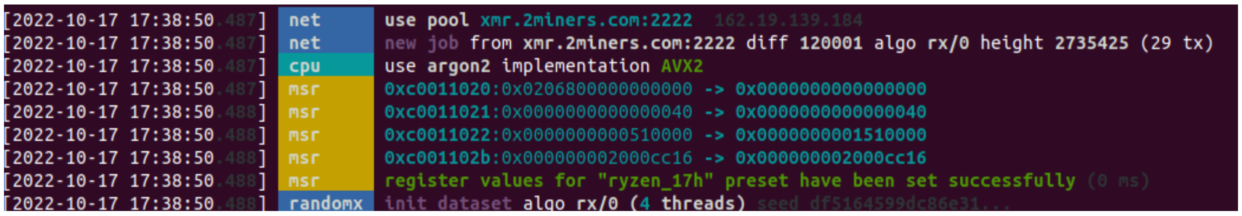
XMRig shares the CPU with connection pool as below. The mining pool is general pool hosted in the public domain and anyone can join by specifying the pool main at the launch of process.



```
- CONNECTION
* pool address      xmr.2miners.com:2222 (162.19.139.184)
* algorithm         rx/0
* difficulty        120001
* connection time   28s
```

Figure 4.5
Intel V-Tune Profile – Miners Pool

Model Specific Registers are the configurations register to control the default settings of CPU, by altering the values via MSR, the flow can be controlled. The default settings for hardware prefetcher are altered by XMRig process to increase the hash rate.



```
[2022-10-17 17:38:50.487]  net      use pool xmr.2miners.com:2222  162.19.139.184
[2022-10-17 17:38:50.487]  net      new job from xmr.2miners.com:2222 diff 120001 algo rx/0 height 2735425 (29 tx)
[2022-10-17 17:38:50.487]  cpu      use argon2 implementation AVX2
[2022-10-17 17:38:50.487]  msr      0xc0011020:0x0206800000000000 -> 0x0000000000000000
[2022-10-17 17:38:50.488]  msr      0xc0011021:0x0000000000000040 -> 0x0000000000000040
[2022-10-17 17:38:50.488]  msr      0xc0011022:0x0000000000510000 -> 0x0000000001510000
[2022-10-17 17:38:50.488]  msr      0xc001102b:0x000000002000cc16 -> 0x000000002000cc16
[2022-10-17 17:38:50.488]  msr      register values for "ryzen_17h" preset have been set successfully (0 ms)
[2022-10-17 17:38:50.488]  randomx  init dataset algo rx/0 (4 threads) seed df5164599dc86e31...
```

Figure 4.6
Intel V-Tune Profile – XMRig Model Specific Registers

Hash rate variance seen after the MSR altering settings by XMRig crypto process.

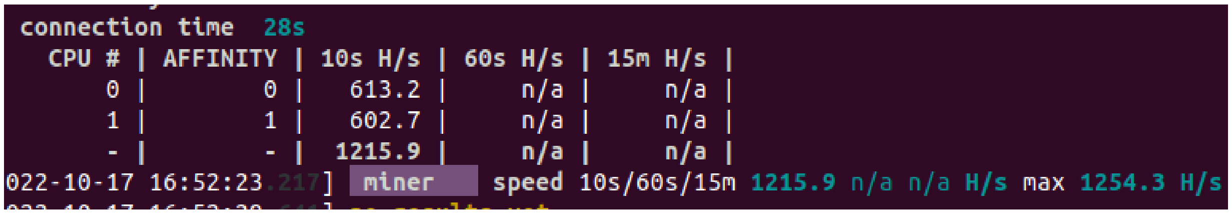

```
 connection time  28s
   CPU # | AFFINITY | 10s H/s | 60s H/s | 15m H/s |
       0 |        0 |   613.2 |     n/a |     n/a |
       1 |        1 |   602.7 |     n/a |     n/a |
       - |        - |  1215.9 |     n/a |     n/a |
022-10-17 16:52:23.217]  miner    speed 10s/60s/15m 1215.9 n/a n/a H/s max 1254.3 H/s
```

Figure 4.7
XMRig Hash Rate

3) Start the monitor of XMRig process via Intel V -Tune in Hot Spot mode

Intel V-Tune tool has several options as follows

vtune <-action> [-action-option] [-global-option] [[--] <target> [target-options]]

vtune - The name of the V-Tune Profiler command line tool.

<-action> - The action to perform, such as collect or report.

[-action-option] - Action-options modify behavior specific to the action. You can have multiple action-options per action. Using an action-option that does not apply to the action results in a usage error.

[-global-option] - Global-options modify behavior in the same manner for all actions. You can have multiple global-options per action.

[--] <target> - The target application to analyze.

-collect <analysis_type> - hotspots

Identify your most time-consuming source code using one of the available collection modes:

-knob sampling-mode=sw (former Basic Hotspots) to collect hotspots and stack information based on the user-mode sampling and tracing, which does not required sampling drivers but incurs higher collection overhead). This mode cannot be used to profile a system, but must either launch an application/process or attach to one.

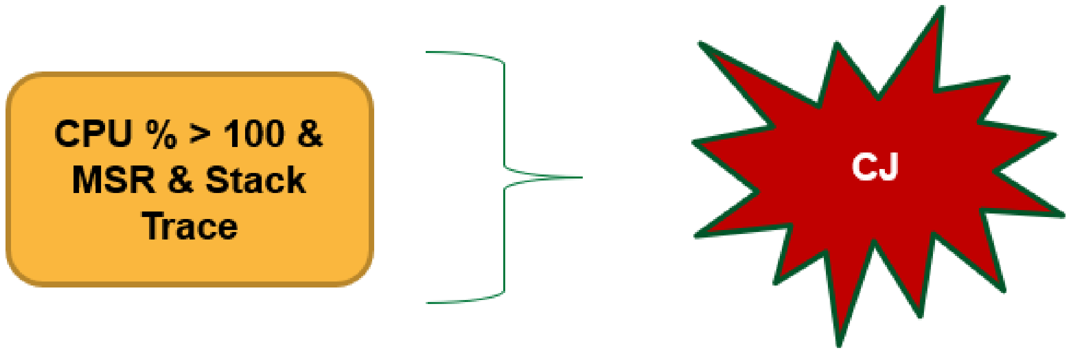
Generate a hotspots collection based on the process id by following Intel V-Tune cmd

```
vtune -collect hotspots -knob sampling-mode=sw -call-stack-mode all  -target-
process xmrig
```

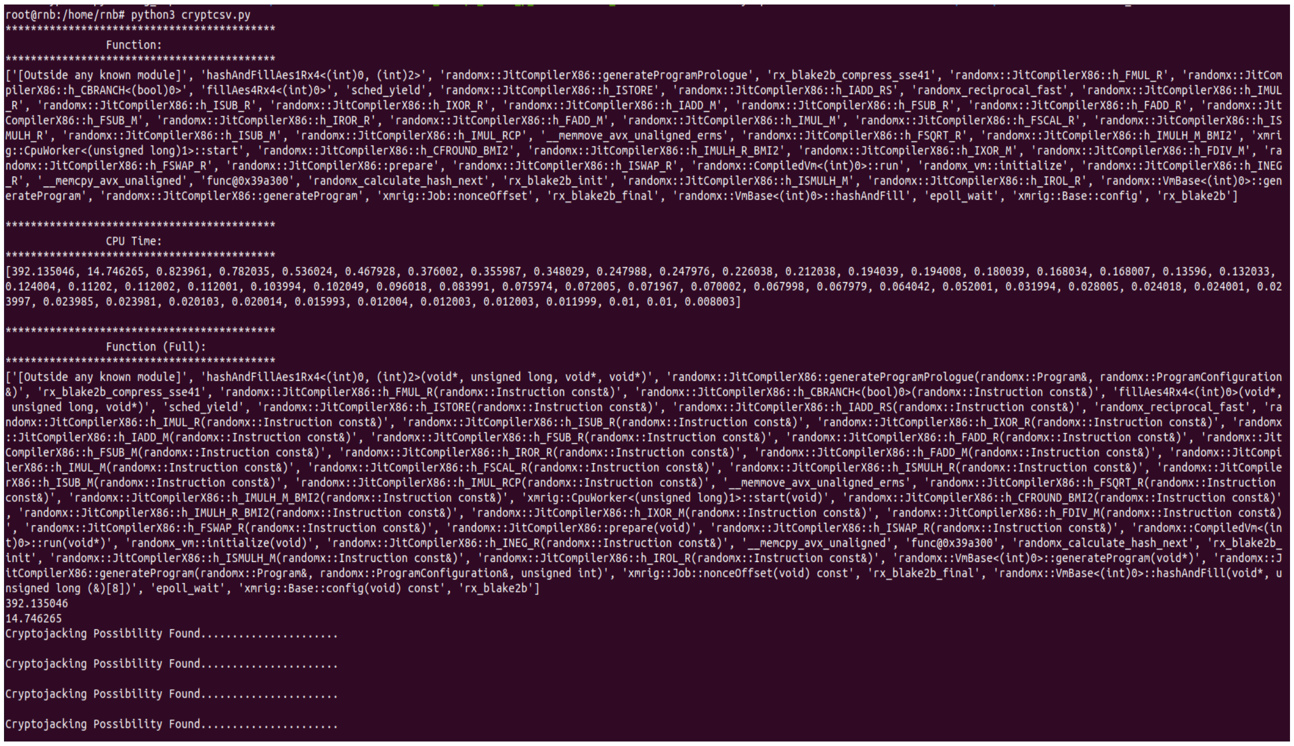
Generate a hotspots report from a hotspots analysis that was previously generated in the directory r0001hs

```
vtune -R hotspots -report-output xmrig.csv -format csv -csv-delimiter comma -r
r0001hs
```

4) Deciding logic for Cryptojacking detection is done by the following formula



CPU Threshold Usage > 100% + MSR value altered for hash rate + Stack trace showing the signature of mining application

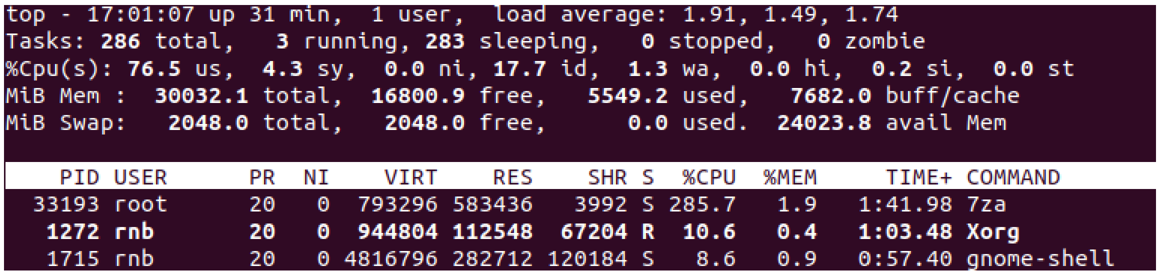


```
root@rnb:/home/rnb# python3 cryptcsv.py
*******************************************
                Function:
*******************************************
['[Outside any known module]', 'hashAndFillAes1Rx4<(int)0, (int)2>', 'randomx::JitCompilerX86::generateProgramPrologue', 'rx_blake2b_compress_sse41', 'randomx::JitCompilerX86::h_FMUL_R', 'randomx::JitCom
pilerX86::h_CBRANCH<(bool)0>', 'fillAes4Rx4<(int)0>', 'sched_yield', 'randomx::JitCompilerX86::h_ISTORE', 'randomx::JitCompilerX86::h_IADD_RS', 'randomx_reciprocal_fast', 'randomx::JitCompilerX86::h_IMUL
_R', 'randomx::JitCompilerX86::h_ISUB_R', 'randomx::JitCompilerX86::h_IXOR_R', 'randomx::JitCompilerX86::h_IADD_M', 'randomx::JitCompilerX86::h_FSUB_R', 'randomx::JitCompilerX86::h_FADD_R', 'randomx::Jit
CompilerX86::h_FSUB_M', 'randomx::JitCompilerX86::h_IROR_R', 'randomx::JitCompilerX86::h_FADD_M', 'randomx::JitCompilerX86::h_IMUL_M', 'randomx::JitCompilerX86::h_FSCAL_R', 'randomx::JitCompilerX86::h_IS
MULH_R', 'randomx::JitCompilerX86::h_ISUB_M', 'randomx::JitCompilerX86::h_IMUL_RCP', '__memmove_avx_unaligned_erms', 'randomx::JitCompilerX86::h_FSQRT_R', 'randomx::JitCompilerX86::h_IMULH_M_BMI2', 'xmri
g::CpuWorker<(unsigned long)1>::start', 'randomx::JitCompilerX86::h_CFROUND_BMI2', 'randomx::JitCompilerX86::h_IMULH_R_BMI2', 'randomx::JitCompilerX86::h_IXOR_M', 'randomx::JitCompilerX86::h_FDIV_M', 'ra
ndomx::JitCompilerX86::h_FSWAP_R', 'randomx::JitCompilerX86::prepare', 'randomx::JitCompilerX86::h_ISWAP_R', 'randomx::CompiledVm<(int)0>::run', 'randomx_vm::initialize', 'randomx::JitCompilerX86::h_INEG
_R', '__memcpy_avx_unaligned', 'func@0x39a300', 'randomx_calculate_hash_next', 'rx_blake2b_init', 'randomx::JitCompilerX86::h_ISMULH_M', 'randomx::JitCompilerX86::h_IROL_R', 'randomx::VmBase<(int)0>::gen
erateProgram', 'randomx::JitCompilerX86::generateProgram', 'xmrig::Job::nonceOffset', 'rx_blake2b_final', 'randomx::VmBase<(int)0>::hashAndFill', 'epoll_wait', 'xmrig::Base::config', 'rx_blake2b']

*******************************************
                CPU Time:
*******************************************
[392.135046, 14.746265, 0.823961, 0.782035, 0.536024, 0.467928, 0.376002, 0.355987, 0.348029, 0.247988, 0.247976, 0.226038, 0.212038, 0.194039, 0.194008, 0.180039, 0.168034, 0.168007, 0.13596, 0.132033,
0.124004, 0.11202, 0.112002, 0.112001, 0.103994, 0.102049, 0.096018, 0.083991, 0.075974, 0.072005, 0.071967, 0.070002, 0.067998, 0.067979, 0.064042, 0.052001, 0.031994, 0.028005, 0.024018, 0.024001, 0.02
3997, 0.023985, 0.023981, 0.020103, 0.020014, 0.015993, 0.012004, 0.012003, 0.012003, 0.011999, 0.01, 0.01, 0.008003]

*******************************************
                Function (Full):
*******************************************
['[Outside any known module]', 'hashAndFillAes1Rx4<(int)0, (int)2>(void*, unsigned long, void*, void*)', 'randomx::JitCompilerX86::generateProgramPrologue(randomx::Program&, randomx::ProgramConfiguration
&)', 'rx_blake2b_compress_sse41', 'randomx::JitCompilerX86::h_FMUL_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_CBRANCH<(bool)0>(randomx::Instruction const&)', 'fillAes4Rx4<(int)0>(void*,
 unsigned long, void*)', 'sched_yield', 'randomx::JitCompilerX86::h_ISTORE(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IADD_RS(randomx::Instruction const&)', 'randomx_reciprocal_fast', 'ra
ndomx::JitCompilerX86::h_IMUL_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_ISUB_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IXOR_R(randomx::Instruction const&)', 'randomx
::JitCompilerX86::h_IADD_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FSUB_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FADD_R(randomx::Instruction const&)', 'randomx::Jit
CompilerX86::h_FSUB_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IROR_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FADD_M(randomx::Instruction const&)', 'randomx::JitCompi
lerX86::h_IMUL_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FSCAL_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_ISMULH_R(randomx::Instruction const&)', 'randomx::JitCompile
rX86::h_ISUB_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IMUL_RCP(randomx::Instruction const&)', '__memmove_avx_unaligned_erms', 'randomx::JitCompilerX86::h_FSQRT_R(randomx::Instruction
const&)', 'randomx::JitCompilerX86::h_IMULH_M_BMI2(randomx::Instruction const&)', 'xmrig::CpuWorker<(unsigned long)1>::start(void)', 'randomx::JitCompilerX86::h_CFROUND_BMI2(randomx::Instruction const&)'
, 'randomx::JitCompilerX86::h_IMULH_R_BMI2(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IXOR_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FDIV_M(randomx::Instruction const&)
', 'randomx::JitCompilerX86::h_FSWAP_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::prepare(void)', 'randomx::JitCompilerX86::h_ISWAP_R(randomx::Instruction const&)', 'randomx::CompiledVm<(in
t)0>::run(void*)', 'randomx_vm::initialize(void)', 'randomx::JitCompilerX86::h_INEG_R(randomx::Instruction const&)', '__memcpy_avx_unaligned', 'func@0x39a300', 'randomx_calculate_hash_next', 'rx_blake2b_
init', 'randomx::JitCompilerX86::h_ISMULH_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IROL_R(randomx::Instruction const&)', 'randomx::VmBase<(int)0>::generateProgram(void*)', 'randomx::J
itCompilerX86::generateProgram(randomx::Program&, randomx::ProgramConfiguration&, unsigned int)', 'xmrig::Job::nonceOffset(void) const', 'rx_blake2b_final', 'randomx::VmBase<(int)0>::hashAndFill(void*, u
nsigned long (&)[8])', 'epoll_wait', 'xmrig::Base::config(void) const', 'rx_blake2b']
392.135046
14.746265
Cryptojacking Possibility Found.....................

Cryptojacking Possibility Found.....................

Cryptojacking Possibility Found.....................

Cryptojacking Possibility Found.....................
```

Figure 4.8
Intel V-Tune Profile – XMRig Decider Logic

5) Non-Crypto Process Analysis 7zip

7zip is the general file compression utility available on all OS flavors (Linux, Microsoft etc.). The purpose of 7zip is to show the CPU consumption that can be compared to the crypto process.



```
top - 17:01:07 up 31 min,  1 user,  load average: 1.91, 1.49, 1.74
Tasks: 286 total,   3 running, 283 sleeping,   0 stopped,   0 zombie
%Cpu(s): 76.5 us,  4.3 sy,  0.0 ni, 17.7 id,  1.3 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :  30032.1 total,  16800.9 free,   5549.2 used,   7682.0 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.  24023.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  33193 root      20   0  793296 583436   3992 S 285.7   1.9   1:41.98 7za
   1272 rnb       20   0  944804 112548  67204 R  10.6   0.4   1:03.48 Xorg
   1715 rnb       20   0 4816796 282712 120184 S   8.6   0.9   0:57.40 gnome-shell
```

Figure 4.9
Intel V-Tune Profile – 7za CPU Utilization

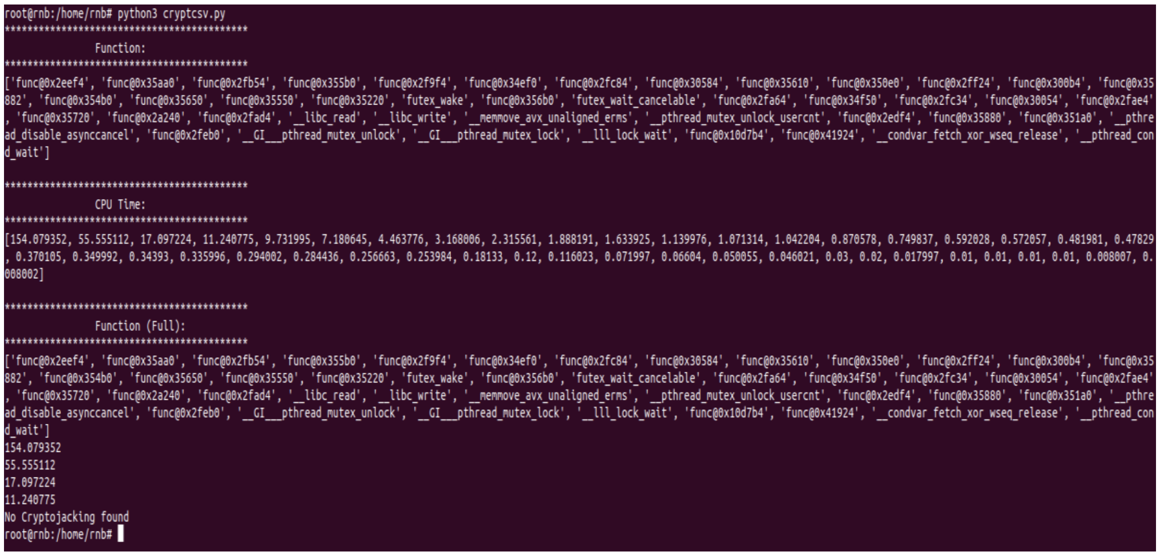
Generate a hotspots collection based on the process id by following Intel V-Tune cmd

vtune -collect hotspots -knob sampling-mode=sw -call-stack-mode all -target-process 7za

vtune -R hotspots -report-output 7za.csv -format csv -csv-delimiter comma -r r0002hs

Deciding logic for Non Cryptojacking detection

CPU Threshold Usage > 100% + MSR value altered for hash rate + Stack trace showing the signature of compression application



```
root@rnb:/home/rnb# python3 cryptcsv.py
*******************************************
                Function:
*******************************************
['func@0x2eef4', 'func@0x35aa0', 'func@0x2fb54', 'func@0x355b0', 'func@0x2f9f4', 'func@0x34ef0', 'func@0x2fc84', 'func@0x30584', 'func@0x35610', 'func@0x350e0', 'func@0x2ff24', 'func@0x300b4', 'func@0x35
882', 'func@0x354b0', 'func@0x35650', 'func@0x35550', 'func@0x35220', 'futex_wake', 'func@0x356b0', 'futex_wait_cancelable', 'func@0x2fa64', 'func@0x34f50', 'func@0x2fc34', 'func@0x30054', 'func@0x2fae4'
, 'func@0x35720', 'func@0x2a240', 'func@0x2fad4', '__libc_read', '__libc_write', '__memmove_avx_unaligned_erms', '__pthread_mutex_unlock_usercnt', 'func@0x2edf4', 'func@0x35880', 'func@0x351a0', '__pthre
ad_disable_asynccancel', 'func@0x2feb0', '__GI___pthread_mutex_unlock', '__GI___pthread_mutex_lock', '__lll_lock_wait', 'func@0x10d7b4', 'func@0x41924', '__condvar_fetch_xor_wseq_release', '__pthread_con
d_wait']

*******************************************
                CPU Time:
*******************************************
[154.079352, 55.555112, 17.097224, 11.240775, 9.731995, 7.180645, 4.463776, 3.168006, 2.315561, 1.888191, 1.633925, 1.139976, 1.071314, 1.042204, 0.870578, 0.749837, 0.592028, 0.572057, 0.481981, 0.47829
, 0.370105, 0.349992, 0.34393, 0.335996, 0.294002, 0.284436, 0.256663, 0.253984, 0.18133, 0.12, 0.116023, 0.071997, 0.06604, 0.050055, 0.046021, 0.03, 0.02, 0.017997, 0.01, 0.01, 0.01, 0.01, 0.008007, 0.
008002]

*******************************************
                Function (Full):
*******************************************
['func@0x2eef4', 'func@0x35aa0', 'func@0x2fb54', 'func@0x355b0', 'func@0x2f9f4', 'func@0x34ef0', 'func@0x2fc84', 'func@0x30584', 'func@0x35610', 'func@0x350e0', 'func@0x2ff24', 'func@0x300b4', 'func@0x35
882', 'func@0x354b0', 'func@0x35650', 'func@0x35550', 'func@0x35220', 'futex_wake', 'func@0x356b0', 'futex_wait_cancelable', 'func@0x2fa64', 'func@0x34f50', 'func@0x2fc34', 'func@0x30054', 'func@0x2fae4'
, 'func@0x35720', 'func@0x2a240', 'func@0x2fad4', '__libc_read', '__libc_write', '__memmove_avx_unaligned_erms', '__pthread_mutex_unlock_usercnt', 'func@0x2edf4', 'func@0x35880', 'func@0x351a0', '__pthre
ad_disable_asynccancel', 'func@0x2feb0', '__GI___pthread_mutex_unlock', '__GI___pthread_mutex_lock', '__lll_lock_wait', 'func@0x10d7b4', 'func@0x41924', '__condvar_fetch_xor_wseq_release', '__pthread_con
d_wait']
154.079352
55.555112
17.097224
11.240775
No Cryptojacking found
root@rnb:/home/rnb# ▌
```

Figure 4.10
Intel V-Tune Profile – 7za Decider Logic

The idea of malware detection in the run time production platforms poses the challenges of performance degradation to the user accessing the application. Our research study proposes adapting Istio (Istio) framework in analysing the logs by diverting some production traffic to Intel V-Tune. By this the user will not experience the performance degradation.

Istio's traffic routing rules let you easily control the flow of traffic and API calls between services. Istio simplifies configuration of service-level properties like circuit breakers, timeouts, and retries, and makes it easy to set up important tasks like A/B testing, canary rollouts, and staged rollouts with percentage-based traffic splits. It also provides out-of-box reliability features that help make your application more resilient against failures of dependent services or the network.

Istio's traffic (Traffic shifting) shifting can be configured by two Istio Custom Resources, namely Destination Rule and Virtual Service. In short, in a Destination Rule so called subsets need to be defined to identify specific versions of a service, and in a Virtual Service different weights can be assigned to these subsets to control how much traffic should be directed to each service version.

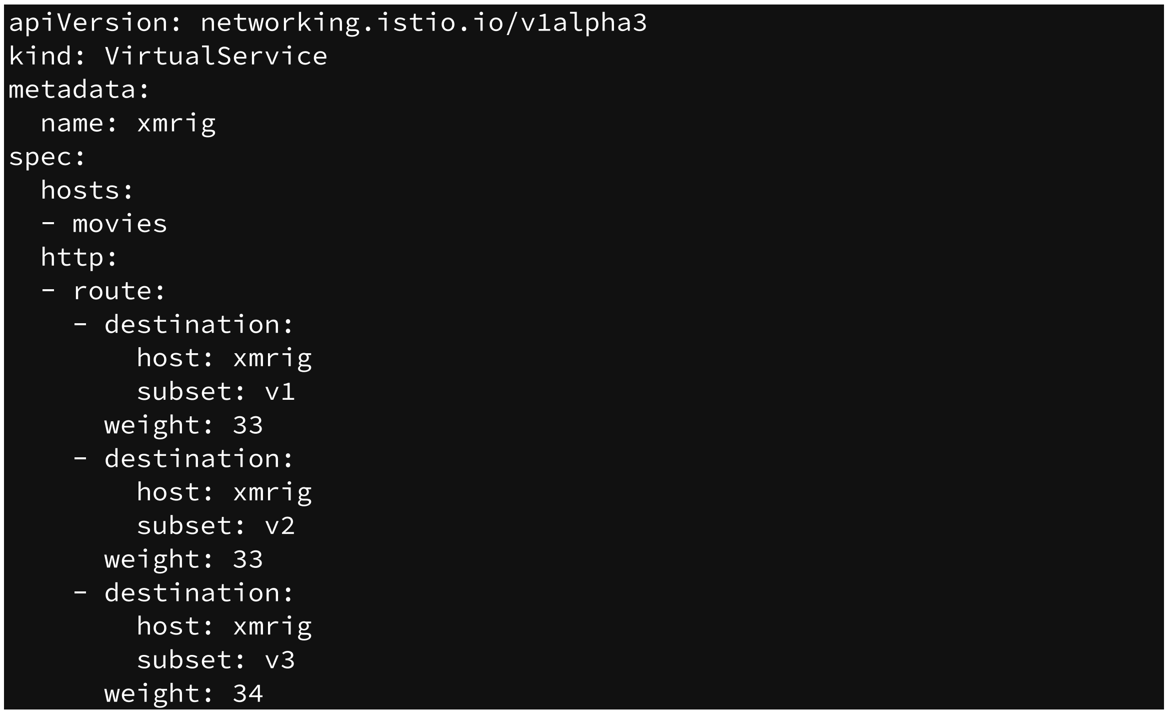
For the subsets, you’ll need to define a Destination Rule like this:

```
kind: DestinationRule
metadata:
  name: xmrig
spec:
  host: xmrig
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version:
v3
```

Figure 4.11
Istio – Destination Rules

This defines the following subset names: v1, v2 and v3. These subset names can be used later in Virtual Services to route traffic to the appropriate version.

For the weights, need to define a Virtual Service like this:



```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: xmrig
spec:
  hosts:
  - movies
  http:
  - route:
    - destination:
        host: xmrig
        subset: v1
      weight: 33
    - destination:
        host: xmrig
        subset: v2
      weight: 33
    - destination:
        host: xmrig
        subset: v3
      weight: 34
```

Figure 4.12
Istio – Virtual Service

Now the Virtual Service is set to route 33% of all traffic to subset v1, 33% to v2 and 34% to v3.

4.4 Summary

This research goal is to identify and evaluate the K8S resources affected by the crypto miner's malware. For this the researcher identified the open-source tool Intel V-Tune profiler to monitor the miner process. The report or logs generated out will be transformed to the CSV format for further analysis by the decider logic.

Above scenario also discussed the non-miner process 7zip application which is meant for compressing/decompressing the files. The idea behind choosing is to make sure the CPU usage overshoots the threshold value which will help to analyze the core logic of crypto mining process.

The latest report by (scmagazine, 2022) shows how easily hackers are targeting the dockers, K8S resources.

The next chapter will discuss the major findings and provide some practical results of the research and finally make recommendations for future research.

CHAPTER V:
DISCUSSION, IMPLICATIONS AND RECOMMENDATIONS

5.1 Introduction

The primary objective of this study is to identify the customer application hardware resource consumption beyond the threshold value in the Public or Private cloud environment, Private Data canters and any small and medium business IT centers managing the hardware clusters. The research has utilized the open-source tools and framework for the demonstration of the pipeline discussed in previous section.

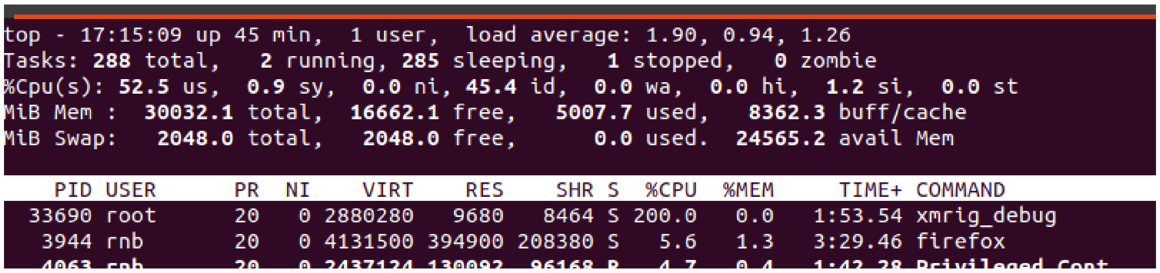
The findings from the previous chapter demonstrate that the phenomena, opinions and lived experiences of researcher in the world of cyber threats.

5.2 Discussion of Results

Following topic discuss the results comparison between the crypto application vs non crypto application. Here “XMRig” is the open-source crypto mining application chosen to compare 7zip application.

5.2.1 XMRig Crypto Application Analysis:

Measure the CPU utilization by the XMRig process

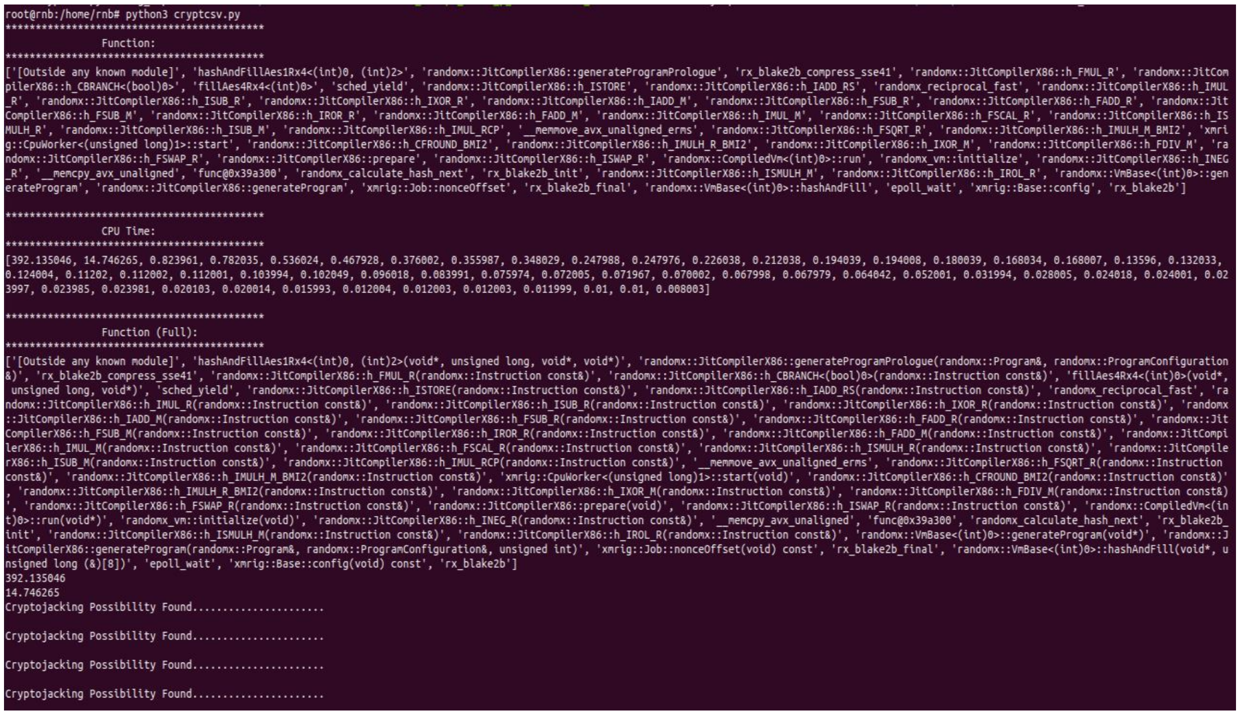


```
top - 17:15:09 up 45 min,  1 user,  load average: 1.90, 0.94, 1.26
Tasks: 288 total,   2 running, 285 sleeping,   1 stopped,   0 zombie
%Cpu(s): 52.5 us,  0.9 sy,  0.0 ni, 45.4 id,  0.0 wa,  0.0 hi,  1.2 si,  0.0 st
MiB Mem :  30032.1 total,  16662.1 free,   5007.7 used,   8362.3 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.  24565.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  33690 root      20   0 2880280   9680   8464 S 200.0   0.0   1:53.54 xmrig_debug
   3944 rnb       20   0 4131500 394900 208380 S   5.6   1.3   3:29.46 firefox
```

Figure 5.1
Intel V-Tune Profile – XMRig CPU Utilization

Deciding logic for Cryptojacking detection

CPU Threshold Usage > 100% + MSR value altered for hash rate + Stack trace showing the signature of mining application

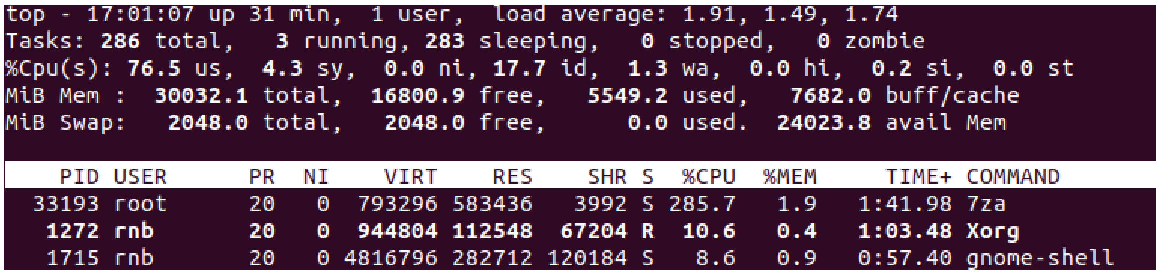


```
root@rnb:/home/rnb# python3 cryptcsv.py
********************************************
              Function:
********************************************
['[Outside any known module]', 'hashAndFillAes1Rx4<(int)0, (int)2>', 'randomx::JitCompilerX86::generateProgramPrologue', 'rx_blake2b_compress_sse41', 'randomx::JitCompilerX86::h_FMUL_R', 'randomx::JitCom
pilerX86::h_CBRANCH<(bool)0>', 'fillAes4Rx4<(int)0>', 'sched_yield', 'randomx::JitCompilerX86::h_ISTORE', 'randomx::JitCompilerX86::h_IADD_RS', 'randomx_reciprocal_fast', 'randomx::JitCompilerX86::h_IMUL
_R', 'randomx::JitCompilerX86::h_ISUB_R', 'randomx::JitCompilerX86::h_IXOR_R', 'randomx::JitCompilerX86::h_IADD_M', 'randomx::JitCompilerX86::h_FSUB_R', 'randomx::JitCompilerX86::h_FADD_R', 'randomx::Jit
CompilerX86::h_FSUB_M', 'randomx::JitCompilerX86::h_IROR_R', 'randomx::JitCompilerX86::h_FADD_M', 'randomx::JitCompilerX86::h_IMUL_M', 'randomx::JitCompilerX86::h_FSCAL_R', 'randomx::JitCompilerX86::h_IS
MULH_R', 'randomx::JitCompilerX86::h_ISUB_M', 'randomx::JitCompilerX86::h_IMUL_RCP', '__memmove_avx_unaligned_erms', 'randomx::JitCompilerX86::h_FSQRT_R', 'randomx::JitCompilerX86::h_IMULH_M_BMI2', 'xmri
g::CpuWorker<(unsigned long)1>::start', 'randomx::JitCompilerX86::h_CFROUND_BMI2', 'randomx::JitCompilerX86::h_IMULH_R_BMI2', 'randomx::JitCompilerX86::h_IXOR_M', 'randomx::JitCompilerX86::h_FDIV_M', 'ra
ndomx::JitCompilerX86::h_FSWAP_R', 'randomx::JitCompilerX86::prepare', 'randomx::JitCompilerX86::h_ISWAP_R', 'randomx::CompiledVm<(int)0>::run', 'randomx_vm::initialize', 'randomx::JitCompilerX86::h_INEG
_R', '__memcpy_avx_unaligned', 'func@0x39a300', 'randomx_calculate_hash_next', 'rx_blake2b_init', 'randomx::JitCompilerX86::h_ISMULH_M', 'randomx::JitCompilerX86::h_IROL_R', 'randomx::VmBase<(int)0>::gen
erateProgram', 'randomx::JitCompilerX86::generateProgram', 'xmrig::Job::nonceOffset', 'rx_blake2b_final', 'randomx::VmBase<(int)0>::hashAndFill', 'epoll_wait', 'xmrig::Base::config', 'rx_blake2b']

********************************************
              CPU Time:
********************************************
[392.135046, 14.746265, 0.823961, 0.782035, 0.536024, 0.467928, 0.376002, 0.355987, 0.348029, 0.247988, 0.247976, 0.226038, 0.212038, 0.194039, 0.194008, 0.180039, 0.168034, 0.168007, 0.13596, 0.132033,
0.124004, 0.11202, 0.112002, 0.112001, 0.103994, 0.102049, 0.096018, 0.083991, 0.075974, 0.072005, 0.071967, 0.070002, 0.067998, 0.067979, 0.064042, 0.052001, 0.031994, 0.028005, 0.024018, 0.024001, 0.02
3997, 0.023985, 0.023981, 0.020103, 0.020014, 0.015993, 0.012004, 0.012003, 0.012003, 0.011999, 0.01, 0.01, 0.008003]

********************************************
              Function (Full):
********************************************
['[Outside any known module]', 'hashAndFillAes1Rx4<(int)0, (int)2>(void*, unsigned long, void*, void*)', 'randomx::JitCompilerX86::generateProgramPrologue(randomx::Program&, randomx::ProgramConfiguration
&)', 'rx_blake2b_compress_sse41', 'randomx::JitCompilerX86::h_FMUL_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_CBRANCH<(bool)0>(randomx::Instruction const&)', 'fillAes4Rx4<(int)0>(void*,
 unsigned long, void*)', 'sched_yield', 'randomx::JitCompilerX86::h_ISTORE(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IADD_RS(randomx::Instruction const&)', 'randomx_reciprocal_fast', 'ra
ndomx::JitCompilerX86::h_IMUL_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_ISUB_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IXOR_R(randomx::Instruction const&)', 'randomx
::JitCompilerX86::h_IADD_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FSUB_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FADD_R(randomx::Instruction const&)', 'randomx::Jit
CompilerX86::h_FSUB_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IROR_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FADD_M(randomx::Instruction const&)', 'randomx::JitCompi
lerX86::h_IMUL_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FSCAL_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_ISMULH_R(randomx::Instruction const&)', 'randomx::JitCompile
rX86::h_ISUB_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IMUL_RCP(randomx::Instruction const&)', '__memmove_avx_unaligned_erms', 'randomx::JitCompilerX86::h_FSQRT_R(randomx::Instruction
const&)', 'randomx::JitCompilerX86::h_IMULH_M_BMI2(randomx::Instruction const&)', 'xmrig::CpuWorker<(unsigned long)1>::start(void)', 'randomx::JitCompilerX86::h_CFROUND_BMI2(randomx::Instruction const&)'
, 'randomx::JitCompilerX86::h_IMULH_R_BMI2(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IXOR_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_FDIV_M(randomx::Instruction const&)
', 'randomx::JitCompilerX86::h_FSWAP_R(randomx::Instruction const&)', 'randomx::JitCompilerX86::prepare(void)', 'randomx::JitCompilerX86::h_ISWAP_R(randomx::Instruction const&)', 'randomx::CompiledVm<(in
t)0>::run(void*)', 'randomx_vm::initialize(void)', 'randomx::JitCompilerX86::h_INEG_R(randomx::Instruction const&)', '__memcpy_avx_unaligned', 'func@0x39a300', 'randomx_calculate_hash_next', 'rx_blake2b_
init', 'randomx::JitCompilerX86::h_ISMULH_M(randomx::Instruction const&)', 'randomx::JitCompilerX86::h_IROL_R(randomx::Instruction const&)', 'randomx::VmBase<(int)0>::generateProgram(void*)', 'randomx::J
itCompilerX86::generateProgram(randomx::Program&, randomx::ProgramConfiguration&, unsigned int)', 'xmrig::Job::nonceOffset(void) const', 'rx_blake2b_final', 'randomx::VmBase<(int)0>::hashAndFill(void*, u
nsigned long (&)[8])', 'epoll_wait', 'xmrig::Base::config(void) const', 'rx_blake2b']
392.135046
14.746265
Cryptojacking Possibility Found.....................

Cryptojacking Possibility Found.....................

Cryptojacking Possibility Found.....................

Cryptojacking Possibility Found.....................
```

Figure 5.2
Intel V-Tune Profile – XMRig Cryptojacking detection

5.2.3 Non-Crypto Process 7zip Analysis:

Non-Crypto Process Analysis 7zip (Compressing application)

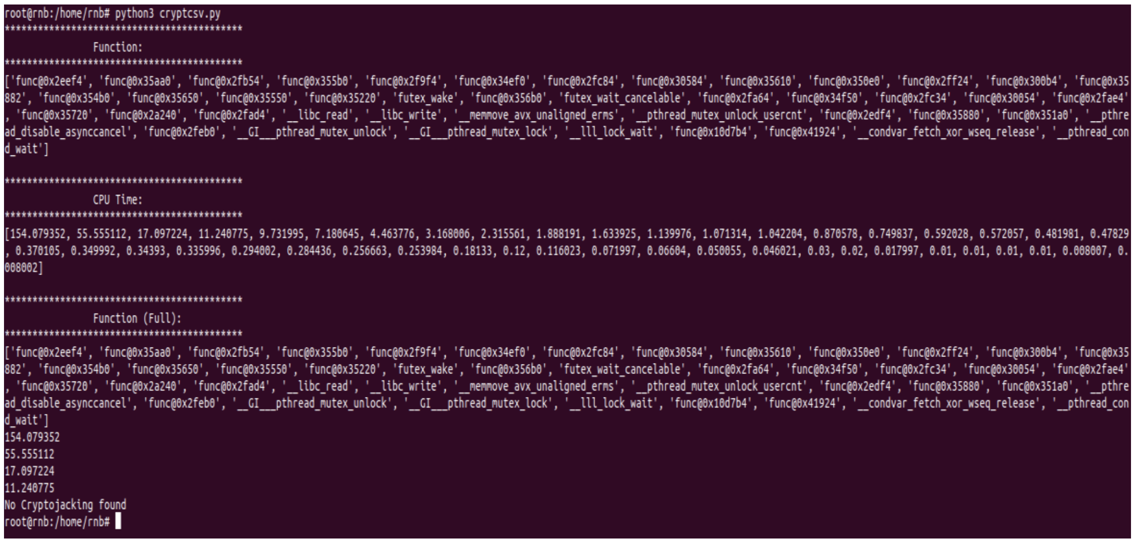


```
top - 17:01:07 up 31 min,  1 user,  load average: 1.91, 1.49, 1.74
Tasks: 286 total,   3 running, 283 sleeping,   0 stopped,   0 zombie
%Cpu(s): 76.5 us,  4.3 sy,  0.0 ni, 17.7 id,  1.3 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :  30032.1 total,  16800.9 free,   5549.2 used,   7682.0 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.  24023.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  33193 root      20   0  793296 583436   3992 S 285.7   1.9   1:41.98 7za
   1272 rnb       20   0  944804 112548  67204 R  10.6   0.4   1:03.48 Xorg
   1715 rnb       20   0 4816796 282712 120184 S   8.6   0.9   0:57.40 gnome-shell
```

Figure 5.3
Intel V-Tune Profile 7za CPU Utilization

Deciding logic for Cryptojacking detection

CPU Threshold Usage > 100% + MSR value altered for hash rate + Stack trace showing the signature of compression application



```
root@rnb:/home/rnb# python3 cryptcsv.py
*******************************************
              Function:
*******************************************
['func@0x2eef4', 'func@0x35aa0', 'func@0x2fb54', 'func@0x355b0', 'func@0x2f9f4', 'func@0x34ef0', 'func@0x2fc84', 'func@0x30584', 'func@0x35610', 'func@0x350e0', 'func@0x2ff24', 'func@0x300b4', 'func@0x35
882', 'func@0x354b0', 'func@0x35650', 'func@0x35550', 'func@0x35220', 'futex_wake', 'func@0x356b0', 'futex_wait_cancelable', 'func@0x2fa64', 'func@0x34f50', 'func@0x2fc34', 'func@0x30054', 'func@0x2fae4'
, 'func@0x35720', 'func@0x2a240', 'func@0x2fad4', '__libc_read', '__libc_write', '__memmove_avx_unaligned_erms', '__pthread_mutex_unlock_usercnt', 'func@0x2edf4', 'func@0x35880', 'func@0x351a0', '__pthre
ad_disable_asynccancel', 'func@0x2feb0', '__GI___pthread_mutex_unlock', '__GI___pthread_mutex_lock', '__lll_lock_wait', 'func@0x10d7b4', 'func@0x41924', '__condvar_fetch_xor_wseq_release', '__pthread_con
d_wait']

*******************************************
              CPU Time:
*******************************************
[154.079352, 55.555112, 17.097224, 11.240775, 9.731995, 7.180645, 4.463776, 3.168006, 2.315561, 1.888191, 1.633925, 1.139976, 1.071314, 1.042204, 0.870578, 0.749837, 0.592028, 0.572057, 0.481981, 0.47829
, 0.370105, 0.349992, 0.34393, 0.335996, 0.294002, 0.284436, 0.256663, 0.253984, 0.18133, 0.12, 0.116023, 0.071997, 0.06604, 0.050055, 0.046021, 0.03, 0.02, 0.017997, 0.01, 0.01, 0.01, 0.01, 0.008007, 0.
008002]

*******************************************
              Function (Full):
*******************************************
['func@0x2eef4', 'func@0x35aa0', 'func@0x2fb54', 'func@0x355b0', 'func@0x2f9f4', 'func@0x34ef0', 'func@0x2fc84', 'func@0x30584', 'func@0x35610', 'func@0x350e0', 'func@0x2ff24', 'func@0x300b4', 'func@0x35
882', 'func@0x354b0', 'func@0x35650', 'func@0x35550', 'func@0x35220', 'futex_wake', 'func@0x356b0', 'futex_wait_cancelable', 'func@0x2fa64', 'func@0x34f50', 'func@0x2fc34', 'func@0x30054', 'func@0x2fae4'
, 'func@0x35720', 'func@0x2a240', 'func@0x2fad4', '__libc_read', '__libc_write', '__memmove_avx_unaligned_erms', '__pthread_mutex_unlock_usercnt', 'func@0x2edf4', 'func@0x35880', 'func@0x351a0', '__pthre
ad_disable_asynccancel', 'func@0x2feb0', '__GI___pthread_mutex_unlock', '__GI___pthread_mutex_lock', '__lll_lock_wait', 'func@0x10d7b4', 'func@0x41924', '__condvar_fetch_xor_wseq_release', '__pthread_con
d_wait']
154.079352
55.555112
17.097224
11.240775
No Cryptojacking found
root@rnb:/home/rnb# 
```

Figure 5.4
Intel V-Tune Decider Logic

5.2.3 CPU Threshold Comparison

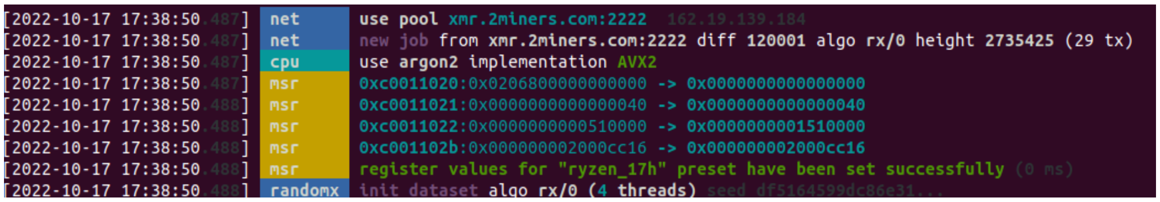
The following table compares the CPU threshold vs actual recorded value for crypto application process (XMRig) and non-crypto application process (7zip).

Process Name	CPU Threshold Limit in %	Actual Recorded Value in %
XMRig (Crypto Application)	**100**	**200**
7zip (Non-Crypto Application)	**100**	**285.7**

Table 5.1
Intel V-Tune CPU Threshold

From the above table it is clear that CPU threshold will be the first parameter to be considered by the Crypto logic decider process. The threshold value plus the stack trace of the Intel V- Tune will help in checking the miner’s logic.

Also, the Model specific registers settings of default values vs XMRig process settings on platform specific will pinpoint the intension of the crypto miner's logic.



```
[2022-10-17 17:38:50.487]  net      use pool xmr.2miners.com:2222  162.19.139.184
[2022-10-17 17:38:50.487]  net      new job from xmr.2miners.com:2222 diff 120001 algo rx/0 height 2735425 (29 tx)
[2022-10-17 17:38:50.487]  cpu      use argon2 implementation AVX2
[2022-10-17 17:38:50.487]  msr      0xc0011020:0x0206800000000000 -> 0x0000000000000000
[2022-10-17 17:38:50.488]  msr      0xc0011021:0x0000000000000040 -> 0x0000000000000040
[2022-10-17 17:38:50.488]  msr      0xc0011022:0x0000000000510000 -> 0x0000000001510000
[2022-10-17 17:38:50.488]  msr      0xc001102b:0x000000002000cc16 -> 0x000000002000cc16
[2022-10-17 17:38:50.488]  msr      register values for "ryzen_17h" preset have been set successfully (0 ms)
[2022-10-17 17:38:50.488]  randomx  init dataset algo rx/0 (4 threads) seed df5164599dc86e31...
```

Figure 5.5
Intel V-Tune MSR Values

The MSR value is changed by the XMRig miners process is specifically changed to increase the Hash rate by the miners. This is done to avoid the hardware prefetching feature that results in poor hash rate by RandomX algorithm. This plays a vital clue for the Crypto logic decider to conclude the mining.

Supported CPUs:

- **Intel** (Nehalem, Westmere, Sandy Bridge, Ivy Bridge, Haswell, Broadwell and newer)
- **Ryzen** (All Zen based CPUs: Ryzen, Threadripper, EPYC)

After launching the XMRig application, it modifies the MSR value to bypass the hardware prefetching capability. If the application is exited for some reason for error or on graceful exit, the initial value of the MSR registers is restored.

Preset values shipped with the XMRig miner application:

Following values are preset via the MSR when the XMRig application is launched.

- `"wrmsr": ["0x1a4:0xf"]`

  Intel CPU preset value to bypass hardware prefetch capability.

- `"wrmsr": ["0xc0011020:0x4480000000000",` `"0xc0011021:0x1c000200000040:0xffffffffffffffdf",` `"0xc0011022:0xc000000401500000", "0xc001102b:0x2000cc14"]`

  AMD Ryzen (Zen3) CPU ‘s preset value to bypass hardware prefetch capability.

- `"wrmsr": ["0xc0011020:0x0", "0xc0011021:0x40:0xffffffffffffffdf",` `"0xc0011022:0x1510000", "0xc001102b:0x2000cc16"]`

  AMD Ryzen (Zen1/Zen2) CPU’s preset value to bypass hardware prefetch capability.

- `"wrmsr": ["0xc0011020:0x0", "0xc0011021:0x60",` `"0xc0011022:0x510000", "0xc001102b:0x1808cc16"]`

  Special custom value for first generation Zen CPUs (with known hardware

  bug) to disable opcache and enable MSR mod.

5.2.4 Research Implications on Business

As part of the research, following business sectors and many others are direct beneficiary of eliminating the cryptojacking but not limited to.

Area
Financial Sector:
Healthcare Sector
IT and Technology Sector
E-commerce and Retail Sector:
Education Sector
Energy Sector

Business Benefit	Examples
Preserved Computing Resources	An e-commerce platform experiences faster page load times, leading to higher conversion rates and improved user satisfaction.
Enhanced Security and Reputation	A financial institution strengthens its cybersecurity measures, protecting customer data and enhancing its reputation as a trusted institution.
Cost Savings	A cloud service provider reduces energy costs and avoids hardware degradation by removing cryptojacking, resulting in significant cost savings.
Maintained Customer Trust	A software company promptly patches vulnerabilities exploited for cryptojacking, earning customer trust and loyalty, reducing churn rates.
Avoided Legal and Regulatory Consequences	An online gaming platform eliminates cryptojacking to comply with legal requirements, avoiding regulatory fines and maintaining its standing in the industry.
Improved Business Continuity	A healthcare provider removes cryptojacking from its network, ensuring uninterrupted critical systems and preventing harm caused by disruptions.

Table 5.2
Business Benefits Components

Also, the following use case demonstrates the approximate potential savings that cloud service provider can benefits that hosts 200 server nodes.

Components	Potential Earnings/Savings	Potential Cost Savings for 200 nodes (Approximate Estimates)
Reduced Energy Consumption	Savings on electricity costs due to decreased energy usage.	$200K/year
Extended Hardware Lifespan	Reduced expenses on hardware maintenance and replacement.	$50K/ year
Improved Efficiency and Performance	Enhanced resource allocation and improved customer experience.	$60K/year
Maintenance and Incident Response	Decreased expenses associated with incident response, system recovery, and customer support.	$30K/year
Preserved Reputation and Trust	Retained customers, increased customer acquisition, and improved brand reputation.	N/A (Indirect Benefit)

Table 5.3
Components for TOC Gain

General benefits according to aquasec (2004) that any sector or company can get are summarized as below :

Preserved Computing Resources:

By preventing cryptojacking, businesses can ensure that their computing resources are used for intended purposes. This preserves system performance, improves productivity, and avoids unnecessary strain on hardware components.

Enhanced Security and Data Protection:

Avoiding cryptojacking helps businesses protect their systems, networks, and sensitive data. It reduces the risk of unauthorized access, data breaches, and potential legal and regulatory consequences associated with compromised security.

Improved Operational Efficiency:

Without the burden of cryptojacking incidents, businesses can operate with greater efficiency. Systems run at optimal performance, enabling smoother workflows, faster response times, and increased productivity.

Cost Savings:

By avoiding cryptojacking, businesses can save costs associated with increased energy consumption, hardware damage, and incident response. They can allocate resources more effectively, reduce operational expenses, and optimize budget allocation.

Maintained Customer Trust:

By prioritizing security and protecting customer data, businesses can maintain customer trust and loyalty. Avoiding cryptojacking incidents demonstrates a commitment to customer privacy and helps build a positive reputation.

Uninterrupted Business Continuity:

Cryptojacking can disrupt operations, leading to downtime and financial losses. By proactively preventing these incidents, businesses can ensure uninterrupted services, maintain customer satisfaction, and avoid reputational damage.

Regulatory Compliance:

Cryptojacking incidents can violate data protection and privacy regulations. By avoiding such incidents, businesses can demonstrate compliance with relevant laws, ensuring the protection of customer information and avoiding potential legal consequences.

Positive Brand Image:

Taking a proactive stance against cryptojacking demonstrates responsible business practices. It enhances a business's brand image, positioning it as a trusted and security-conscious organization within the industry and among customers.

According to Aulia Arif Wardana et al (2019) and research methods can determine the benefits to general management by the IT team in adopting the steps for eliminating the Cryptojacking.

1. **Focus on Technical Risk Mitigation**: Frame the discussion around the technical risks associated with cryptojacking. Explain the potential impact on computing resources, system performance, and data integrity. Emphasize that eliminating cryptojacking is crucial for maintaining operational continuity, safeguarding critical systems, and protecting sensitive data from unauthorized access.

2. **Quantify Technical Implications**: Provide technical insights and statistics to demonstrate the potential impact of cryptojacking incidents. Estimate the potential strain on computing resources, the energy consumption of mining activities, and the potential hardware degradation caused by prolonged cryptojacking. Present these figures alongside the projected technical improvements achievable by implementing preventive measures, showcasing the technical return on investment (ROI).

3. **Link to Security Compliance and Best Practices**: Explain how cryptojacking incidents can violate security best practices and compliance requirements. Highlight the importance of adhering to security frameworks, such as the NIST Cybersecurity Framework or ISO 27001, and how implementing preventive

measures aligns with these frameworks. Emphasize the technical and regulatory risks of non-compliance and potential financial and legal consequences.

4. **Illustrate Operational Efficiency Gains**: Showcase how eliminating cryptojacking can enhance operational efficiency from a technical perspective. Discuss the benefits of improved system performance, optimized resource allocation, and reduced incidents of system downtime. Demonstrate how proactive measures, such as implementing intrusion detection systems, enhancing network segmentation, and leveraging threat intelligence, can contribute to increased productivity, streamlined workflows, and improved technical operations.

5. **Highlight Technical Security:** Measures: Emphasize the significance of implementing technical security measures to maintain customer trust and protect the company's brand reputation. Explain that by eliminating cryptojacking, the company showcases its commitment to secure computing practices and customer privacy. Illustrate how this commitment enhances technical operations, mitigates risks, and establishes the company as a leader in the industry.

6. **Present ML/AI Solutions**: Discuss the advantages of leveraging ML/AI solutions for cryptojacking detection and prevention. Explain how ML/AI algorithms can

analyze large datasets, identify patterns, and detect cryptojacking activities in real-time. Highlight the technical advantages of adaptive ML/AI models that continuously learn and improve their detection capabilities, staying ahead of evolving threats. Present technical details on the implementation and integration of ML/AI solutions within the existing infrastructure.

7. **Provide Technical Implementation Plan**: Outline a technical implementation plan for adopting steps to eliminate cryptojacking. Present a roadmap that includes necessary technical upgrades, software/hardware configurations, integration with existing security systems, and employee training on security practices. Address technical concerns, such as infrastructure compatibility, scalability, and performance impact, and propose feasible technical solutions.

5.3 Recommendations for Future Research

As stated, there are some limitations to this study, which future researchers may wish to explore further.

The research intends to claim that with the advanced in the computational capacity of the Cloud resources and the efficiency of the Intel V-Tune Profiler together helps to minimize the threats by the malware or hackers' application in utilizing the HW resources in the K8S environment.

This research study claims the reducing the run time of threat detection of an existing application in the customer environment. The window period of threat detection will be reduced as its makes use of the micro service architecture out of which part of the logs collection will happen in run time.

The K8S is chosen as the reference orchestrating platform to deploy containers. The solution is equally applicable for other orchestrating platforms such as Docker Swarm, Rancher, Red hat OpenShift, Mesos, AWS, Microsoft Azure, GKE etc.

A study on the new and upcoming hardware platforms and new crypto miner's applications will always lead to reduce the crypto miner's target. The decider logic function should be kept on adding to support new crypto miners' logic. Also, to support more hardware architectures, the future versions.

Following are the possible list of open-source miner applications that can be targeted by the academic professionals or enterprise customers to identify the cryptojacking.

- Pionex
- ECOS
- Kryptex Miner
- Cudo Miner
- BeMine
- Awesome Miner
- BFGMiner
- MultiMiner
- EasyMiner
- CGMiner
- BTCMiner
- DiabloMiner
- NiceHash Miner

Following are the possible list of cloud service providers that can be used as the platform to identify the cryptojacking

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud (GCP—formerly Google Cloud Platform)
- IBM Cloud
- Oracle Cloud
- Alibaba Cloud
- RedHat
- Heroku
- Digital Ocean
- CloudFlare
- Linode
- Cloudways
- Rackspace

.

Following are the possible list of service mesh for K8S that can be used as the framework for identify the cryptojacking.

- Linkerd
- AWS App Mesh
- Consul Connect
- Kuma
- Traefik Mesh
- Apache ServiceComb
- Network Service Mesh (NSM)
- Kiali Operator
- NGINX Service Mesh
- Aspen Mesh
- Open Service Mesh (OSM)
- Grey Matter
- OpenShift Service Mesh

According to Kubiya (kubiya.ai), The concept of generative AI describes machine learning algorithms that can create new content from minimal human input. The field has rapidly advanced in the past few years, with projects such as the text authorship tool ChatGPT and realistic image creator DALL-E2 attracting mainstream attention.

Generative AI isn’t just for content creators, though. It’s also poised to transform technical work in the software engineering and DevOps fields. GitHub Copilot, the controversial “AI pair programmer,” is already prompting reconsideration of how code is written, but collaborative AI’s potential remains relatively unexplored in the DevOps arena.

There’s huge potential for generative AI to redefine how Cryptojacking can be identified, in accordance with the above logic discussed and to train the AI model accordingly. Some points below gives a strong contention to choose the generative AI Path as part of the future work to carry on identifying the Cryptojacking.

Automatic Failure Detection, with Suggested Remedies

- ✓ Failures are a constant problem for developers and operators alike. They're unpredictable interruptions that force an immediate context switch to prioritize a fix. This hinders productivity, slows down release schedules, and causes frustration when remedial work doesn't go to plan.

- ✓ AI agents can detect faults and investigate their cause. They can combine their analysis with generative capabilities and knowledge of past failures to suggest immediate actions, within the context where the alert's displayed.

- ✓ Consider a simple Kubernetes example: The assistant notices that production is down; realizes the Pod has been evicted due to resource constraints; and provides action buttons to restart the Pod, scale the cluster, or terminate other disused resources. The team can resolve the incident with a single click, instead of spending several minutes manually troubleshooting.

On-Demand Code/Config Generation and Deployment

- ✓ Generative AI's ability to author code provides incredible value. Layering in conversational intents makes it more accessible and convenient. You can ask an AI agent to set up a new project, config file, or Terraform state definition by writing a brief message into a chat interface. The agent can prompt you to supply values for any template placeholders, then notify appropriate stakeholders that the content's ready for review.

- ✓ After approval's been obtained, AI can inform the original developer, launch the project into a live environment, and provide a link to view the deployment and start iterating upon it. This condenses several distinct sequences into one self-service action for developers. Ops teams don't need to manually provision the project's resources ahead of time, allowing them to stay focused on their own tasks.

- ✓ The next generation of AI agents go beyond simple text and photo creation to support fully automated prompt-driven workflows. Bi-directional AI lets you start processes using natural language to interact with your cloud or other resources. AI doesn't need to be told which platform you're using, or the specific steps it should run.

5.4 Conclusion

The primary research method for this study is literature review and the data collected by the Intel V-Tune Profiler in effectively zeroing the threat models. Various results, and the accuracy has taken into the account as part of the research study. The recent trends show the multifold adaptation of the blockchain technology for all the areas (Finance, Automobile, Healthcare, etc.). The pipeline is fed with data that relates to the problem domain and metadata that attributes a label to the data. The Model will look for the system calls and other HW resource utilization (Hot Spot analysis) usages of the existing application deployed in the safe environments before deploying the applications into the cloud platforms. The various literature concepts and gaps (such has minimized the time in detecting the malware and reducing the storage needed to collect the logs of the application) mentioned have well taken into the account in enhancing the research study.

Using machine learning (ML) and artificial intelligence (AI) for eliminating cryptojacking offers several benefits. ML/AI algorithms can analyze large volumes of data, detect patterns, and identify cryptojacking activities with high accuracy. By leveraging ML/AI, businesses can proactively identify and respond to cryptojacking incidents in real-time, preventing unauthorized mining activities and protecting computing resources. ML/AI-based solutions can adapt to evolving attack techniques, continuously learning from new data, and improving detection capabilities. This advanced technology enables businesses to enhance their cybersecurity posture, preserve system performance, and reduce the risk of data breaches. Furthermore, ML/AI can assist in analyzing network traffic, identifying vulnerabilities, and automatically applying security patches, thereby streamlining incident response, and improving overall operational efficiency. By harnessing the power of ML/AI, businesses can fortify their defenses against cryptojacking, safeguard customer trust, and maintain a secure computing environment.

As stated, following research questions has been answered:

- How can the customer identify the applications targeted by crypto miners to make use of the HW platforms?

As discussed, the E2E pipeline concept which explains the end-to-end scenario starting from the source data collection, data cleaning, report generation and finally analysis.

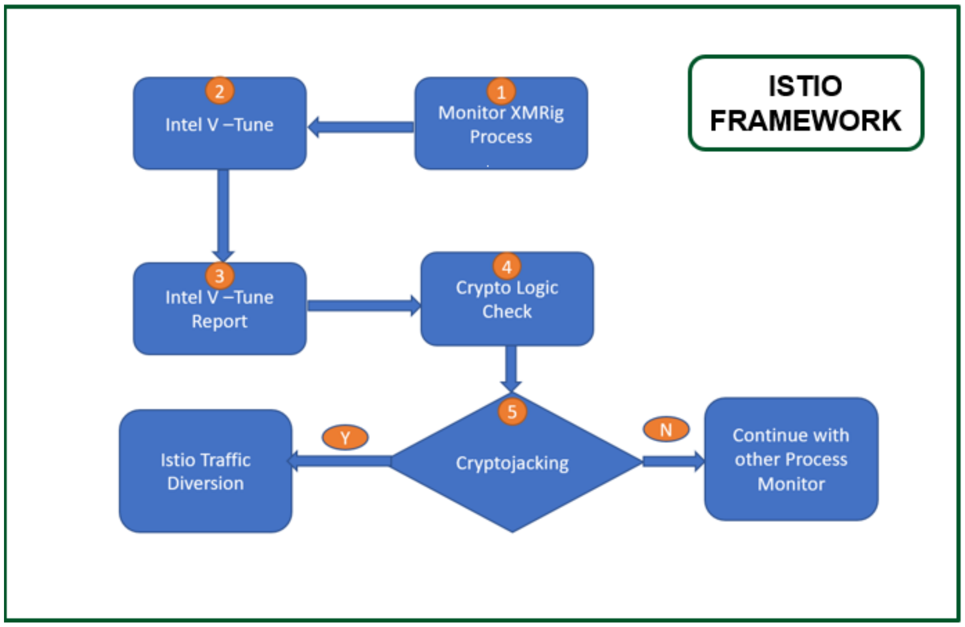


Figure 5.6
E2E Pipeline

The pipeline has been explained in detail in the methodology section and discussed the outcome of the results clearly in the result section. The justification has been provided for the research question.

- Whether the open-source methods/tools like Intel V-Tune profiler can be used to identify?

  The Intel V-Tune profiler is a solid tool in analysing the process on any recommended OS flavours such as Linux, Microsoft OS etc. The pipeline setup proves a solid line of study for system administrators. Although the tool used for this research has targeted X86 based CPU's. This is because most of the deployed data centres servers are X86 based.

- What best resolution methods can be employed to remove the miners targeted application dynamically in the production network?

  The E2E Pipeline is developed on the concept of K8S Mesh service framework. Traffic Re-direction on the fly ensures that there is no disruption in the service to the customers.

The research helps the academic professionals in analyzing the various open source for identifying the cryptojacking. In fact, the various mining applications can be a worth considering using the pipeline suggested.

For the enterprise segment, the research study if the pipeline would help them to uncover the potential threats in their production networks if the tools used efficiently.

The research not only focus the problem of identifying the miners process, but also the suggestion of bringing down the miner’s process dynamically in the production environment helps the customer to bring down the TCO.

APPENDIX A:

CA	COHERENCY AGENT
CFAA	COMPUTER FRAUD AND ABUSE ACT
CJ	CRYPTOJACKING
CPU	CNTRAL PROCESSIGN UNIT
CSR	CONFIGURATION SPACE REGISTER
CSV	COMMA SEPARATED VALUE
DLT	DISTRIBUTED LEDGER TECHNOLOGY
E2E	END-TO-END PIPILINE CYCLE
GDPR	GENERAL DATA PROTECTION REGULATION
ICU	INSTRUCTION CACHE UNIT
ISTIO	KUBERNETS ISTIO
K8S	KUBERNETES
LBR	LAST BRANCH RECORD
MLC	MID LEVEL CACHE
MSR	MODEL SPECIFIC REGISTERS
P2P	PEER-TO-PEER

PIPELINE	PROCESS STARTING FROM DATA COLLECTION TO ANALYSIS
PROCESS	LINUX OS PROCESS
SGX	INTEL SOFTWARE GUARD EXTENSIONS
SKU	STOCK KEEPING UNIT
TOC	TOTAL COST OF OWNERSHIP
V-TUNE	INTEL V-TUNE PROFILER
X86	X86 CPU ARCHITECTURE

BIBLIOGRAPHY

A. A. Rusu et al., “Progressive neural networks,” 2016, arXiv:1606.04671.

A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O’Reilly, “Adversarial deep learning for robust detection of binary encoded malware,” in Proc. IEEE Security Privacy

A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R (2018). Choo,“Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” J. Ambient Intell. Humanized Comput.,

A. C. De Melo 2010, “The new linux ‘perf’ tools,” Slides from Linux Kongress, vol. 18

A. Desnos, E. Petrova, A. Boulgakov, R. Neal, and Z. Mithra (2018), “Flow-graph analysis of system calls for exploit detection,” Tech. Discl. Commons, Jun.

A. J. Elbirt. Fast and efficient implementation of aes via instruction set extensions. In AINAW ’07: Proceedings of the 21st International Confer-

A. P. Bradley, 1997 “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” Pattern Recognit., vol. 30, no. 7, pp. 1145–1159

A. S. Abed, T. C. Clancy, and D. S. Levy (2015), “Applying bag of system calls for anomalous behavior detection of applications in linux containers,” in Proc. IEEE Globecom Workshops

A. Zimba, Z. Wang, M. Mulenga (2020), and N. H. Odongo, “Crypto mining attacks in information systems: An emerging threat to cyber security,” J. Comput. Inf. Syst., vol. 60, pp. 297–308,.

Allagi, S. and Rachh, R., 2019, March. Analysis of Network log data using Machine Learning. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)* (pp. 1-3). IEEE.

Aquasec.com, https://www.aquasec.com/cloud-native-academy/cloud-attacks/cryptojacking/

B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning

Banach, R., 2019, May. Punishment not reward: disincentivising blockchain application misbehaviour. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (pp. 185-187). IEEE.

Buchaca, D., Berral, J.L., Wang, C. and Youssef, A., 2020, October. Proactive container auto-scaling for cloud native machine learning services. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)* (pp. 475-479). IEEE.

Bui, Thanh. Analysis of Docker Security[J]. Computer Science, 2015.

C. Fernando et al.(2017), "PathNet: Evolution channels gradient descent in super neural networks," CoRR

cointelegraph.com, https://cointelegraph.com/news/the-ethics-of-cryptojacking-rampant-malware-or-ad-free-internet

Controllers. https://kubernetes.io/docs/concepts/architecture/controller/

Crypto mining, https://freemanlaw.com/mining-explained-a-detailed-guide-on-how-cryptocurrency-mining-works/

D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu (2012), and E. Kirda, "A quantitative study of accuracy in system callbased malware detection," in Proc. Int. Symp.

Softw. Testing Anal.,

D. Carlin, P. O’Kane, S. Sezer, and J. Burgess(2018), “Detecting cryptomining using dynamic analysis,” in Proc. 16th Annu. Conf. Privacy Secur. Trust,

D. Ceponis and N. Goranin (2019), “Evaluation of deep learning methods efficiency for malicious and benign system calls classification on the AWSCTD,” Secur. Commun.

D. Draghicescu, A. Caranica, A. Vulpe, and O. Fratu, “Cryptomining application fingerprinting method,” in Proc. Int. Conf. Commun., 2018, pp. 543–546.

Davidovici-Nora, M. (2014). Paid and Free Digital Business Models - Innovations in the Video Game Industry. Digiworld Economic Journal, 83-102.

Day, D. V., Gronn, P. & Salas, E. (2004), Leadership capacity in teams, Leadership Quarterly, Elsevier, 15(6), 721-892

De Prato, G., Feijóo, C., & Simon, J.-P. (2014). Innovations in the Video Game Industry: Changing Global Markets. Digiworld Economic Journal, 17-38.

Docker Hub. https://registry.hub.docker.com/

Doug Burger and Todd M. Austin. The simplescalar tool set, version 2.0. SIGARCH Comput. Archit. News, 25(3):13–25, 1997. ISSN 0163-5964.

Dwivedi, R. (2019). Indian Startups: Analyzing Their Vulnerabilities & Prevailing Challenges.

E. Ates et al.(2018), “Taxonomist: Application detection through rich monitoring data,” in Proc. Eur. Conf. Parallel Process.

enterslice.com , https://enterslice.com/learning/cryptojacking-in-banking-industry/#How_to_prevent_Cryptojacking

F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani(2016), “Evaluation of machine learning classifiers for mobile malware detection,” Soft Comput

F. Zenke, B. Poole, and S. Ganguli (2017), “Continual learning through synaptic intelligence,” in Proc. Mach. Learn. Res. Federal Information Processing Standards Publication 197. Specification for the advanced encryption standard (aes), 2001.

Federal Information Processing Standards Publication 46-1. Data encryption standard (des), 1988.

Federal Information Processing Standards Publication 46-3. Data encryption standard (des) - tdea, 1999. for classification of malware system call sequences,” in Proc. Australasian Joint Conf.

G.M. Bertoni, L. Breveglieri, F. Roberto, and F. Regazzoni. Speeding up aes by extending a 32 bit processor instruction set. In Application-specific Systems, Architectures and Processors, 2006. ASAP ’06. International Conference on, pages 275–282, Sept. 2006. doi: 10.1109/ASAP.2006.62.

Gulenko, A., Acker, A., Kao, O. and Liu, F., 2020, August. Ai-governance and levels of automation for aiops-supported system administration. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)* (pp. 1-6). IEEE.

H. Eberle, A. Wander, N. Gura, Sheueling Chang-Shantz, and V. Gupta. Architectural extensions for elliptic curve cryptography over gf(2/sup m/) on 8-bit microprocessors. In Application-Specific Systems, Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on, pages 343–349, 23-25 July 2005. doi: 10.1109/ASAP.2005.15.

H. Liang, Q. Hao, M. Li, and Y. Zhang (2016), "Semantics-based anomaly detection of processes in linux containers," in Proc. Int. Conf. Identification Inf. Knowl. Internet Things

H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, 2017 "A survey on ensemble learning for data stream classification," ACM Comput. Surv., vol. 50, no. 2, 2017, Art. no. 23.

Hans Eberle, Nils Gura, Sheueling Chang Shantz, Vipul Gupta, Leonard Rarick, and Shreyas Sundaram. A public-key cryptographic processor for rsa and ecc. In ASAP '04: Proceedings of the ApplicationSpecific Systems, Architectures and Processors, 15th IEEE International Conference on (ASAP'04), pages 98–110, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2226-2. doi: http://dx.doi.org/10.1109/ASAP.2004.6.

Hyperledger Component, https://blockchain-automation-framework.readthedocs.io/en/latest/architectureref/hyperledger-fabric.html

Hyperledger, hyperledger-fabric.readthedocs.io

I. Branovic, R. Giorgi, and E. Martinelli. A workload characterization of elliptic curve cryptography methods in embedded environments. ACM SIGARCH Computer Architecture News, 32(3):27–34, June 2004. ISSN 0163-5964. doi: http://doi.acm.org/10.1145/1024295.1024299.

I. Kononenko et al., "An efficient explanation of individual classifications using game theory," J. Mach. Learn. Res., vol. 11, no. Jan,

Investopedia, https://www.investopedia.com/terms/b/blockchain.asp

Istio, https://istio.io/latest/docs/

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, 2014 "Empirical evaluation of gated recurrent neural networks on sequence modeling," NIPS Workshop Deep Learn.

J. Kirkpatrick et al (2017)., "Overcoming catastrophic forgetting in neural networks," Proc. Nat. Acad. Sci. United States of America.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, 2015 "Understanding neural networks through deep visualization,"

J. Zhang, K. Zhang, Z. Qin, H. Yin, and Q. Wu, 2018 "Sensitive system calls based packed malware variants detection using principal component initialized multilayers neural networks," Cybersecurity

Jerome Burke, John McDonald, and Todd Austin. Architectural support for fast symmetric-key cryptography. SIGPLAN Not., 35(11):178–189, 2000. ISSN 0362-1340. doi: http://doi.acm.org/10.1145/356989.357006.

Joan Daemen and Vincent Rijmen. The design of Rijndael: AES — the Advanced Encryption Standard. Springer-Verlag, 2002. ISBN 3-540- 42580-2.

KARN ET AL.: CRYPTOMINING DETECTION IN CONTAINER CLOUDS USING SYSTEM CALLS AND EXPLAINABLE MACHINE LEARNING 689[36] D. Stopel and B. Bernstein, "Runtime detection of vulnerabilities in an application layer of software containers,".

Karn, R.R., Kudva, P., Huang, H., Suneja, S. and Elfadel, I.M., 2020. Cryptomining detection in container clouds using system calls and explainable machine learning. *IEEE Transactions on Parallel and Distributed Systems*, *32*(3), pp.674-691.

Kubernetes, https://kubernetes.io/docs/home

Li, H., Zhan, D., Liu, T. and Ye, L., 2019, November. Using deep-learning-based memory analysis for malware detection in cloud. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)* (pp. 1-6). IEEE.

Liz Rice, https://learning.oreilly.com/library/view/container-security/9781492056690/ch07.html#idm45153508247688

M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, 2012 "Spatio-temporal convolutional sparse auto-encoder for sequence classification," in Proc. Brit. Mach. Vis. Conf., 2012, pp. 1–12.

M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu, 2018 "Feedback autonomic provisioning for guaranteeing performance in MapReduce systems," IEEE Trans. Cloud Comput., vol. 6, no. 4, pp. 1004–1016

M. Dimjasevi————————————c, S. Atzeni, I. Ugrina, and Z. Rakamaric (2016), "Evaluation of Android malware detection based on system calls," in Proc. ACM Int. Workshop Secur. Privacy Analytics.

M. Melis, D. Maiorca, B. Biggio, G. Giacinto, and F. Roli(2018),"Explaining black-box android malware detection," in Proc. 26th Eur. Signal Process. Conf.

M. Salehi and M. Amini, “Android malware detection using Markov Chain model of application behaviors in requesting system services,” CoRR, vol. abs/1711.05731, 2017. http://arxiv.org/abs/1711.05731

M. T. Ribeiro, S. Singh, and C. Guestrin,2016 “Why should i trust you?: Explaining the predictions of any classifier,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 1135–1144.

Michael Brown, Darrel Hankerson, Julio L´opez, and Alfred Menezes. Software implementation of the nist elliptic curves over prime fields. In CTRSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology, pages 250–265, London, UK, 2001. Springer-Verlag. ISBN 3-540-41898-9.

MSR, https://www.uptycs.com/blog/cryptominer-elfs-using-msr-to-boost-mining-process
nacdl.org, https://www.nacdl.org/Landing/ComputerFraudandAbuseAct

Nukala, V.S.K.A., 2021. *Website Cryptojacking Detection Using Machine Learning* (Doctoral dissertation, University of Cincinnati). Creswell, J.W. (2014). Research design: Qualitative, quantitative, and mixed methods approaches, 4th edition. Sage Publications.

O. Suciu, S. Coull, and J. Johns, 2019 “Exploring adversarial examples in malware detection,” IEEE Secur. Privacy Workshops, pp. 8–14, May 2019, doi: 10.1109/SPW.2019.00015.

P. Dimotikalis(2016), “Memory forensics and bitcoin mining malware,” Int. Hellenic Univ. Repository.

P. Mishra, K. Khurana, S. Gupta, and M. K. Sharma(2019), “VMAnalyzer: Malware semantic analysis using integrated CNN and bi-directional LSTM for detecting VM-level attacks in cloud," in Proc. 12th Int. Conf. Contemporary Comput.

Paul G. Comba. Exponentiation cryptosystems on the IBM PC. IBM Systems Journal, 29(4):526–538, 1990. [11] Inc. Counterpane Internet Security. The blowfish encryption algorithm, 1993. URL http://www.counterpane.com/blowfish.html.

Pothula, D.R., Kumar, K.M. and Kumar, S., 2019, October. Run Time Container Security Hardening Using A Proposed Model Of Security Control Map. In *2019 Global Conference for Advancement in Technology (GCAT)* (pp. 1-6). IEEE.
pp. 1–18, 2010.

Q. Zhang, 2015 "Modern models for learning large-scale highly skewed online advertising data," Dept. Statist., 2015.

Quantitative data , https://www.questionpro.com/blog/quantitative-data/
Quantitative data , https://www.questionpro.com/blog/quantitative-data/
R. Tahir et al.(2017), "Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises," in Proc. Int. Symp. Res. Attacks Intrusions Defenses

R. Tahir et al., “Mining on someone else’s dime: Mitigating covert mining operations in clouds and enterprises,” in Proc. Int. Symp. Res. Attacks Intrusions Defenses, 2017, pp. 287–310.

R. Tahir, M. Caesar, A. Raza, M. Naqvi, and F. Zaffar (2017), “An anomaly detection fabric for clouds based on collaborative VM communities,” in Proc. 17th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput., 2017, pp. 431–441.

Robert K. Yin. (2014). Case study research design and methods.

Rovnyagin, M.M., Hrapov, A.S., Guminskaia, A.V. and Orlov, A.P., 2020, January. Ml-based heterogeneous container orchestration architecture. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* (pp. 477-481). IEEE.

S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller, 2017“Sequence to sequence autoencoders for unsupervised representation learning from audio,” in Proc. DCASE Workshop, 2017, pp. 17–21.

S. Arnautov et al.(2016), “SCONE: Secure linux containers with intel SGX,” in Proc. 12th USENIX Conf. Operating Syst. Des. Implementation.

S. Bartolini, I. Branovic, R. Giorgi, and E. Martinelli. A performance evaluation of arm isa extension for elliptic curve cryptography over binary finite fields. In Computer Architecture and High-Performance Computing, 2004. SBAC-PAD 2004. 16th

Symposium on, pages 238–245, 27-29 Oct. 2004. doi: 10.1109/SBAC-PAD.2004.5.

S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4MalDroid (2016): A deep learning framework for Android malware detection based on linux kernel system call graphs," in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Workshops

S. M. Lundberg, G. G. Erion, and S.-I. Lee, 2018, "Consistent individualized feature attribution for tree ensembles,

S. M. Vieira, U. Kaymak, and J. M. Sousa, 2010 "Cohen's kappa coefficient as a performance measure for feature selection," in Proc. Int. Conf. Fuzzy Syst.

S. Mousavi, A. Mosavi, A. R. V arkonyi-Koczy, and G. Fazekas, 2017 "Dynamic resource allocation in cloud computing," Acta Polytechnica Hungarica, vol. 14, no. 4, pp. 83–104

S. S. Haykin et al.2009, Neural Networks and Learning Machines/Simon Haykin. New York, NY, USA: Prentice Hall

scmagazine, https://www.scmagazine.com/analysis/cloud-security/researchers-uncover-cryptojacking-campaign-targeting-docker-kubernetes-cloud-servers
Services in Kubernetes. https://kubernetes.io/zh/docs/concepts/servicesnetworking/service/

Silverman, D. (2011) Interpreting Qualitative Data. A Guide to the Principles of Qualitative Research

Sotamaa, O., & Karppi, T. (2010). Games as Services. Tampere: Department of Information Studies and Interactive Media.

T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan (2004), "Ngram-based detection of new malicious code," in Proc. 28th Annu. Int. Comput. Softw. Appl. Conf.

T. Chen and C. Guestrin, 2016 "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining

Tanana, D., 2020, May. Behavior-based detection of cryptojacking malware. In *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)* (pp. 0543-0545). IEEE.

Taylor, C. (2009). The Six Principles of Digital Advertising. International Journal of Advertising, 28(3), 411–418

Taylor, S. J. & Bogdan, R. (1998). Introduction to qualitative research methods: A guidebook and resource (3rd ed.). New York: John Wiley & Sons.

Taylor-Powell, E. & Renner, M. (2003). Analyzing Qualitative Data. University of Wisconsin Cooperative Extension.

Teece, D. J. (2010). Business models, business strategy and innovation. Long range planning, 43(2), 172- 194.

Traffic shifting, https://banzaicloud.com/blog/istio-traffic-shifting/

Tunde-Onadele, O., He, J., Dai, T. and Gu, X., 2019, June. A study on container vulnerability exploit detection. In *2019 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 121-127). IEEE.

Uptcs, https://www.uptycs.com/blog/cryptominer-elfs-using-msr-to-boost-mining-process

V. Ivancevic, N. Igic, B. Terzic, M. Kne zevic, and I. Lukovic 2016, "Decision trees as readable models for early childhood caries," in Proc. Intell. Decision Technol., 2016, pp. 441–451.

W. Ma, P. Duan, S. Liu, G. Gu, and J.-C. Liu (2012), "Shadow attacks: Automatically evading system-call-behavior based malware detection," J. Comput. Virology

W. Wang, B. Ferrell, X. Xu, K. W. Hamlen, and S. Hao (2018), “SEISMIC:Secure in-lined script monitors for interrupting cryptojacks,” in Proc. Eur. Symp. Res. Comput. Secur

W. Yang, D. Kong, T. Xie, and C. A. Gunter, “Malware detection in adversarial settings: Exploiting feature evolutions and confusions in Android apps,” in Proc. 33rd Annu. Comput. Secur. Appl.

Workshops, 2018, pp. 76–82.

X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah (2019), “Android malware detection based on system call sequences and LSTM,” Multimedia Tools Appl., vol. 78, no. 4, pp. 3979–3999

X. Xiao, Z. Wang, Q. Li, S. Xia, and Y. Jiang (2016), “Back-propagationneural network on Markov Chains from system call sequences: A new approach for detecting android malware with system call sequences,” IET Inf. Secur., vol. 11, no. 1, pp. 8–15

Y. Wang and F. Tian, “Recurrent residual learning for sequence classification,2016” in Proc. Conf. Empir. Methods Natural Lang. Process., 2016, pp. 938–943.

Y. Zhang, S. Wang, P. Phillips, and G. Ji, 2014 “Binary PSO with mutation operator for feature selection using decision tree applied to spam detection,” Knowl.-Based Syst., vol. 64, pp. 22–31

Yulianto, Arief & Sukarno, Parman & Wardana, Aulia & Makky, Muhammad. (2019). Mitigation of Cryptojacking Attacks Using Taint Analysis. 234-238. 10.1109/ICITISEE48480.2019.9003742