University of Plymouth

School of Engineering,

Computing, and Mathematics

PRCO304

Final Stage Computing Project

2019/2020

Hardware Assisted Virtualisation Detection

James Murray

BSc (Hons) Computer and Information Security

# Acknowledgements

I would like to thank my supervisor, Marco Palomino, for providing invaluable guidance in the areas of planning, and ethical approval. Without which the breadth of this project would have reduced, affecting the significance of results.

I would also like to thank the team behind “Detecting Hardware-Assisted Virtualization” Michael Brengel, Michael Backes, and Christian Rossow, for releasing their paper on this subject and inspiring me to complete this project.

Finally, I would like to thank my friends and family for supporting me through my studies. Carrying me through difficult times, and keeping my spirits high throughout my university education, despite facing several sources of adversity.

# Abstract

This project is multi-disciplinary, utilising aspects of computer science, security, cloud computing, and data science. The project seeks to investigate correlations between the timing of NOP and CPUID calls, with the enhanced context of CPU model, family, and stepping. This builds upon existing work by (Brengel, et al., 2016) with the goals of verifying the original papers results across heterogenous hardware, discovering the efficacy of hardware virtualisation over time, and producing a neural network to automate virtual environment detection and observation evasion.

The project encompasses multiple phases. Collection, Analysis and Implementation. With deliverables in each. Collection combines computer science and cloud computing to deliver a gathering application which sends test results to a cloud instance. Analysis focuses on collating and visualising the results from the collection stage. Analysis is exploratory and begins to occur during the latter part of the collection phase. Analysis’ results influence the characteristics of the Implementation phase. Implementation delivers a neural network, or neural networks which are trained on the data of the Analysis phase. This deliverable is also tested based upon characteristics observed in the Analysis phase.

Efforts toward individual deliverables are chronicled in the deliverables section, while their results are evaluated in the project results section.

Project management is evaluated in the project management section.

The project as a whole is evaluated in the post-mortem.

The final appendices section contains any supplementary, tangential, or large materials. These are referenced throughout where appropriate.

Table of Contents

[Acknowledgements 2](#_Toc41562250)

[Abstract 2](#_Toc41562251)

[1. Introduction 6](#_Toc41562252)

[1.1 Motivation 6](#_Toc41562253)

[2. Background 6](#_Toc41562254)

[2.1 Existing Research 6](#_Toc41562255)

[2.1.1 Novel Research 7](#_Toc41562256)

[2.1.2 Conclusion 7](#_Toc41562257)

[2.3 Legal, Social, Ethical and Professional Issues 8](#_Toc41562258)

[2.3.1 Legal 8](#_Toc41562259)

[2.3.2 Social and Ethical 8](#_Toc41562260)

[2.3.4 Professional 8](#_Toc41562261)

[3. Method of Approach 9](#_Toc41562262)

[3.1 Project Objectives 9](#_Toc41562263)

[3.2 Requirement Elicitation 9](#_Toc41562264)

[3.2.1 Core (Minimum Viable Product) 9](#_Toc41562265)

[3.2.2 Secondary 9](#_Toc41562266)

[3.2.3 Optional 10](#_Toc41562267)

[3.3 Phased Deliverables 10](#_Toc41562268)

[3.4 Deliverable Interaction 11](#_Toc41562269)

[3.4.1 Gatherer – Collection 11](#_Toc41562270)

[3.4.2 AWS – Collection 11](#_Toc41562271)

[3.4.3 MYSQL - Analysis 11](#_Toc41562272)

[3.4.4 R-Studio – Analysis 11](#_Toc41562273)

[3.4.5 Neural Network - Implementation 11](#_Toc41562274)

[4. Methods and Technologies 12](#_Toc41562275)

[4.1 Methods 12](#_Toc41562276)

[4.1.1 Agile (Scrum) 12](#_Toc41562277)

[4.2 Technologies 13](#_Toc41562278)

[4.2.1 Apache/PHP 13](#_Toc41562279)

[4.2.2 MySQL / Database Development 13](#_Toc41562280)

[5. Deliverables 13](#_Toc41562281)

[5.1 Collection Phase 13](#_Toc41562282)

[5.1.1 Gatherer 13](#_Toc41562283)

[5.1.2 Amazon Web Services Cloud front-end 23](#_Toc41562284)

[5.2 Analysis Phase 24](#_Toc41562285)

[5.2.1 MySQL 24](#_Toc41562286)

[5.2.2 R-Studio 27](#_Toc41562287)

[5.3 Implementation Phase 28](#_Toc41562288)

[5.3.1 Neural Network 28](#_Toc41562289)

[6. Project Results 28](#_Toc41562290)

[6.1 Collection Phase 28](#_Toc41562291)

[6.2 Analysis Phase 29](#_Toc41562292)

[6.2.1 Preliminary Results 29](#_Toc41562293)

[6.2.2 TLB Evictions 30](#_Toc41562294)

[6.2.3 CPUID and Results affected by heavy CPU workload 30](#_Toc41562295)

[6.2.4 Conclusions 34](#_Toc41562296)

[6.3 Implementation Phase 35](#_Toc41562297)

[7. Project Management 35](#_Toc41562298)

[8. Project Post-Mortem 35](#_Toc41562299)

[Implementation vs Minimum Viable Product (overview) 36](#_Toc41562300)

[Successes 36](#_Toc41562301)

[Challenges 36](#_Toc41562302)

[Future improvements 36](#_Toc41562303)

[9. Conclusion 36](#_Toc41562304)

[10. Future research 37](#_Toc41562305)

[Appendices 37](#_Toc41562306)

[Appendix A: User Guide 38](#_Toc41562307)

[Appendix B: Project Management 38](#_Toc41562308)

[Appendix C: Gatherer 38](#_Toc41562309)

[Appendix D: DLL 38](#_Toc41562310)

[Appendix E: AWS 38](#_Toc41562311)

[Appendix F: MySQL 38](#_Toc41562312)

[Appendix G: Initial Testing 38](#_Toc41562313)

[Appendix H: Application for Ethical Approval 38](#_Toc41562314)

# Table of Figures

[Figure 1 Sources of virtualisation detection. Adapted from (Petsas, et al., 2014) 7](#_Toc41562315)

[Figure 2 Phased Deliverables Swimlane diagram 10](#_Toc41562316)

[Figure 3 General Agile Scrum Diagram (Bogazici\_University, 2016) 12](#_Toc41562317)

[Figure 4 Flowchart describing the flow of the program when in use 14](#_Toc41562318)

[Figure 5 Hardware assisted virtualisation diagram (Murray, 2019) 17](#_Toc41562319)

[Figure 6 Simplified diagram of execution within a virtual machine 17](#_Toc41562320)

[Figure 7 Simplified representation of a TLB-Miss and TLB-Hit 18](#_Toc41562321)

[Figure 8 Non-representative visualisation of Virtual and Native CPUID call times 19](#_Toc41562322)

[Figure 9 Visual representation of the Method Three implementation 19](#_Toc41562323)

[Figure 10 Protection rings in x86 CPU architecture (Yeun & Zemerly, 2012) 22](#_Toc41562324)

[Figure 11 Brief overview of cloud infrastructure virtues. Partially adapted from (Pantic & Babar, 2012) and (Tomas & Tordsson, 2013) 23](#_Toc41562325)

[Figure 12 Final EER diagram for the database 25](#_Toc41562326)

[Figure 13 User account write access to the database 26](#_Toc41562327)

[Figure 14 Boxplot comparison of Native(n) and Virtualised(v) CPUID(-C) and NOP(-N) instructions on 3 systems(1-3) 29](file:///C:\Users\James%20Murray\OneDrive\Edumacation\University\Year%206%20Stage%203%202019-2020\PRCO305%2040%20Computing%20Project\HAVD%20Report\Report-Fm.docx#_Toc41562328)

[Figure 15 Method One CPUID/NOP Ratios (Brengel, et al., 2016) 30](file:///C:\Users\James%20Murray\OneDrive\Edumacation\University\Year%206%20Stage%203%202019-2020\PRCO305%2040%20Computing%20Project\HAVD%20Report\Report-Fm.docx#_Toc41562329)

[Figure 16 Method One CPUID/NOP Ratios, Preliminary results 30](file:///C:\Users\James%20Murray\OneDrive\Edumacation\University\Year%206%20Stage%203%202019-2020\PRCO305%2040%20Computing%20Project\HAVD%20Report\Report-Fm.docx#_Toc41562330)

[Figure 17 Zoomed distribution of native CPUID/NOP ratio's under differing system load 31](#_Toc41562331)

[Figure 18 Distribution of Virtualised CPUID/NOP ratio's under differing host system load 32](#_Toc41562332)

[Figure 19 Stealth CPUID calls under differing host system utilisation 33](#_Toc41562333)

[Figure 20 Zoomed Stealth CPUID calls under differing host system utilisation 33](#_Toc41562334)

[Figure 21 Distribution of Stealth CPUID calls under load in a virtual environment 34](#_Toc41562335)

WORD COUNT: 8896

Code Link: <https://github.com/JMurray14/PRCO304-HAVD>

# 1. Introduction

## 1.1 Motivation

Both malware and forensic analyses have changed as a result of the introduction of virtualisation. Discovery techniques have been developed by both malware and forensic developers. These classically consist of environmental awareness techniques utilised in tools such as Scoopy Doo and Jerry (Barrett & Kipper, 2010) now ScoopyNG (Klein, 2008). These tools utilise techniques that are also utilised by malware to gain both debugger awareness and environmental awareness (Chen, et al., 2008). This project will focus on the environmental awareness of malware through the application of Brengel, et al, 2016 to investigate the efficacy of the virtualisation detection.

# 2. Background

The problem posed by virtualisation is the antithesis of it’s purpose. The goal of full virtualisation is to transparently interpose a hypervisor between physical hardware and the virtualised containers so that physical system resources can be shared with multiple containers (Torrey, 2014). The problem encountered is that certain software would like to know the virtualisation status of the environment that they are executed within. While this can be for malicious purposes, such as malware detection evasion, it has implications for digital forensic acquisition and for those administrating distant systems. This project aims to capitalise upon recent research to produce deliverables capable of more accurately distinguishing between virtualised and native execution environments.

## 2.1 Existing Research

Early attempts at virtualisation detection utilise virtualisation artefacts. These artefacts exist due to the implementation of the virtualisation and detecting them allowed reliable detection of the virtualised status of an executing program.

One such older method read descriptor tables, sometimes referred to as vector tables, as signatures. This implementation encountered inaccuracy on certain architectures, and systems with multiple cores (Quist & Smith, 2006).

Another sought the Host-VM communication channels of Modern Virtual Machine Emulators (VME) intended for ease of use. These types of Para virtualised modifications can be detected and interacted with to determine status but can only be exploited when enabled/not mitigated against (Carpenter, et al., 2007).

While many techniques exist to extract artifacts from environments to test for virtualisation, these often are mitigated against to disable their use, and cause unreliable determinations to be made.



Figure 1 Sources of virtualisation detection. Adapted from (Petsas, et al., 2014)

This project focuses on virtualisation overhead measurement present due the hypervisors inspection of certain executed procedures. Previous methodologies have been established to prompt the system for measurable behaviours which are sufficiently different from native execution so as to be clearly distinguishable. One such example relies upon remote host fingerprinting through TCP clock skew inference (Chen, et al., 2008). Relying upon the difference between native (hardware interrupts), and virtualised (software interrupts) clock methods.

Based upon new research, this approach utilises a new variant of overhead measurement, which aims to measure overhead despite any potential measurement mitigations. This would produce a deliverable capable of distinguishing between native and virtual execution despite potential future timing mitigations by hypervisor developers and would ensure its longevity.

### 2.1.1 Novel Research

Brengel et al, puts forward three methodologies for the discovery of virtualised execution. All methods are measurement of virtualisation overhead utilising system derived timing sources. The third method is novel, deriving the timing not from the system directly through reading of the timestamp counter, but through a self-referential source which ties virtualisation performance to the base performance of the system. This is achieved through busy wait loops, and counter increments.

### 2.1.2 Conclusion

While the prima facie of the proposed Stealth CPUID methodology is primitive, the method itself divorces the measurement from timing sources which are vulnerable to tampering by the OS or Hypervisor. With the establishment of discernible timing differences between special and standard instructions, an attacker is able to disregard classic timing sources and rely upon the throughput of this novel method (Brengel, et al., 2016). This provides a resistant path to virtualisation detection and poses a further challenge for malware detection providers utilising automated sandboxing tools. This is suitable for making a virtualisation detection tool, which is resistant to potential future VM exit detection mitigations.

This project will evaluate this novel research and implement the methodologies.

## 2.3 Legal, Social, Ethical and Professional Issues

This project has several important issues that can affect participants and society as a whole.

### 2.3.1 Legal

The need to collect data from heterogenous hardware gives rise to legal issues regarding participation in the collection. The project’s outline requires participants to volunteer their systems information along with test results. The legalities of this involve the GDPR, and the Data Protection Act in the UK.

The particulars which involve this project are the requirements for consent of data sharing. These are that consent need be demonstrated to have occurred, that the obtaining of consent is understood and accessible to the participant, and that the participant can withdraw their data at any time. (Cornock, 2018)

This project partly relies upon Article 17, section 3D of the GDPR wherein processing is still lawful when it is for scientific research and the data minimisation measures taken make removal impossible (PrivacyTrust, 2018). Therefore, in this project the lawfully obtained data can still be used for the purpose for which it is obtained.

Further, this project is able to keep this data due to the entirely anonymous nature of the submissions. Rather than key-coding data based upon personal information, such as with normal studies involving persons, this project key-codes data submissions against CPU identifiers. These CPU identifiers however are entirely based upon the generation and model of CPU and cannot be used to identify an individual participant of the project. This goes beyond the de-identification cited by (Mourby, et al., 2018) and renders the data entirely anonymous, unable to be used to identify individuals.

### 2.3.2 Social and Ethical

Both this project and Brengel et al encounter similar social and ethical concerns. Primarily this research can be used by malicious actors to further the agenda of dynamic analysis evasion. Preventing the automated detection and analysis of certain malware.

This concern is mirrored by the intent to advance the understanding of analysis evasion, to improve analysis tools, and to provoke thought regarding virtualisation implementations.

Overall, this area of study will advance without well intentioned researchers. The importance of fore-knowledge regarding this area is represented in the potential negative products developed behind closed doors. The digital safety of the public should therefore be best protected by pro-active mitigations as a result of the findings in this area.

### 2.3.4 Professional

When releasing software for public use, care must be taken to prevent any potential damage to the systems running the software. Testing has taken place to ensure that the declared unsafe portions of the Gatherer’s code both do not cause memory corruption, and that the memory accessed has been pre-mapped by the application.

# 3. Method of Approach

## 3.1 Project Objectives

To develop several deliverables, which in turn allow the creation of a Neural Network or Neural Network’s that more accurately categorise the programs execution environment as either native or virtual.

* Evaluate Brengel et al
  + Evaluate the main assertions
  + Determine from this research the exact criteria that this project aims to improve upon
* Create a data collection framework for remote submission of data from varied systems
* Analyse this data for behaviours and/or patterns
  + Use these to improve upon Brengel et al
* Create a NN, or NN’s that detect virtual execution at a better rate than Brengel et al
  + Base this upon the data analysis

## 3.2 Requirement Elicitation

Initially elicited from the project’s motivation and base project objectives, these goals gradually mounted throughout development. The initial process of elicitation revealed the base necessities for the project, while further investigation, due to the nature of this exploratory work, produced goals which would enhance aspects of the project.

### 3.2.1 Core (Minimum Viable Product)

* Collection
  + Collect model specific CPU information from CPUID calls
  + Do not collect device/system identifying information
  + Detect virtual execution using the tests of (Brengel, et al., 2016)
  + Send this information to an internet server for collection
  + Not to require elevated or administrative permissions
* Analysis
  + Retrieve this information to local storage
  + Analyse data to determine characteristics
* Implementation
  + Utilise characteristics to build a neural network, or neural networks to determine virtualisation status with at least the efficacy of (Brengel, et al., 2016)

### 3.2.2 Secondary

* Collection
  + Collect CPU microcode revision from CPU
  + Support multiple operating systems
  + Allow for public submissions
  + Correctly apply all methods of (Brengel, et al., 2016)
  + Follow SOLID principles where possible
* Analysis
  + Produce data visualisations that give insights into the phenomena studied
* Implementation
  + Test NN against another set of artificial data created through observation of the analysis phase.
  + Improve upon Brengle et al
    - Method One: Keep True Negative (TN) rate of 100%, and False Positive (FP) rate of 0%.
    - Method One: Improve upon True Positive (TP) rate of 96.6% and False Negative (FN) rate of 3.4%
    - Method Two: Improve upon, TN rate of 98.3%; FP rate of 1.7%; TP rate of 95.2%; and FN rate of 4.8%.
    - Method Three: Improve upon stated TN rate of 99.9%, and TP rate of 99.2%

### 3.2.3 Optional

* Collection
  + Implement auto-scaling and load balanced HTTP solution
* Analysis
  + Discover long term Hardware Virtualization efficiency
  + Discover changes in microcode revision, CPU Stepping, and virtualisation efficiency
* Implementation
  + Train NN to use all methods when discriminating native/virtual execution

A strategy encompassing these, whilst still being malleable to unforeseen requirements, needs a set of deliverables rather than a single work, to be produced.

## 3.3 Phased Deliverables

A screenshot of a cell phone

Description automatically generated

Figure 2 Phased Deliverables Swimlane diagram

In order to achieve the goals of the project a continuous path of information collection, analysis, and implementation is needed. Figure 2 shows the process flow and the inter phase dependencies. This diagram is a high-level but reductive overview of the project within which a minimum viable product or greater, will be developed.

## 3.4 Deliverable Interaction

Interaction between the systems facilitate the later requirements for the deliverables of Analysis and Implementation. These requirements and the inputs and outputs of the deliverables are summarised below.

### 3.4.1 Gatherer – Collection

The Gatherer application is the start of the process responsible for the collection of primary data from disparate systems. It achieves this through the cross platform dotnet core, allowing the program to execute on both target operating systems (Windows and Linux). The gatherer collects the systems CPU information (from CPUID calls) and test results (from the implementation of virtualisation test methodologies). These results are sent to the AWS web service with consent (See Appendix C).

### 3.4.2 AWS – Collection

AWS is the cloud platform which initially hosted the technology stack responsible for data collection. Systems utilising the Linux operating system host Apache for the web front end; MySQL for data storage; and PHP for data processing and response generation (known as a LAMP stack). This deliverable is responsible for data pre-processing (PHP), response generation (PHP), and providing the public face of the project (Apache) (See Appendix E).

Together with the Gatherer, these deliverables compose a loose framework for remote data collection.

This deliverable makes accessible the data needed during the analysis phase.

### 3.4.3 MYSQL - Analysis

This deliverable receives pre-processed data from the PHP portion of the AWS deliverable. This data is stored within the MySQL database, and is retrieved through a migration of the database to local storage. After local storage, it is intended that data which is incongruent with preliminary testing and primary research is separated from the core data before analysis.

This deliverable includes the storage of data, and pre-processing for the analysis deliverable (See Appendix F).

### 3.4.4 R-Studio – Analysis

This deliverable utilises R-Studio to produce visualisation which determine the conditions under which a Neural Network (NN) will be trained and tested. Prior to this, hypotheses for the data have been intuited and are tested to gain further knowledge of the methodologies attributes and value toward the goal of creating a discriminator NN between native and virtual execution.

### 3.4.5 Neural Network - Implementation

The Neural Network (NN) deliverable seeks to create a NN which quickly classifies a system from the minimal usage of the methodologies implemented in the Gatherer deliverable. By using the data visualisations from the previous phase this deliverable can account for the majority scope of values generated by the methodologies and train them into the NN effectively. Through this, previously instruction intensive methodologies could be made to be quiet in terms of suspicious instructions.

# 4. Methods and Technologies

## 4.1 Methods

### 4.1.1 Agile (Scrum)

The primary methodology utilised throughout this project is agile.

Agile provides three main phases, plan, collaboration and deliver. Plan discovers the feature set of a project and first introduces these into a product backlog through user stories. These detail the use of the product and exclaims their needs. This initial stage allows requirements to be envisioned and refined in advance so that during the collaboration phase the correct deliverable is created or worked towards.

Traditionally collaboration encompasses multiple daily scrum meetings, and the sprint of development itself. Sprints are time blocks of arbitrary length focusing on a taskset or feature for development. Sprints include discovery of requirements, design of software architecture, development, and testing.

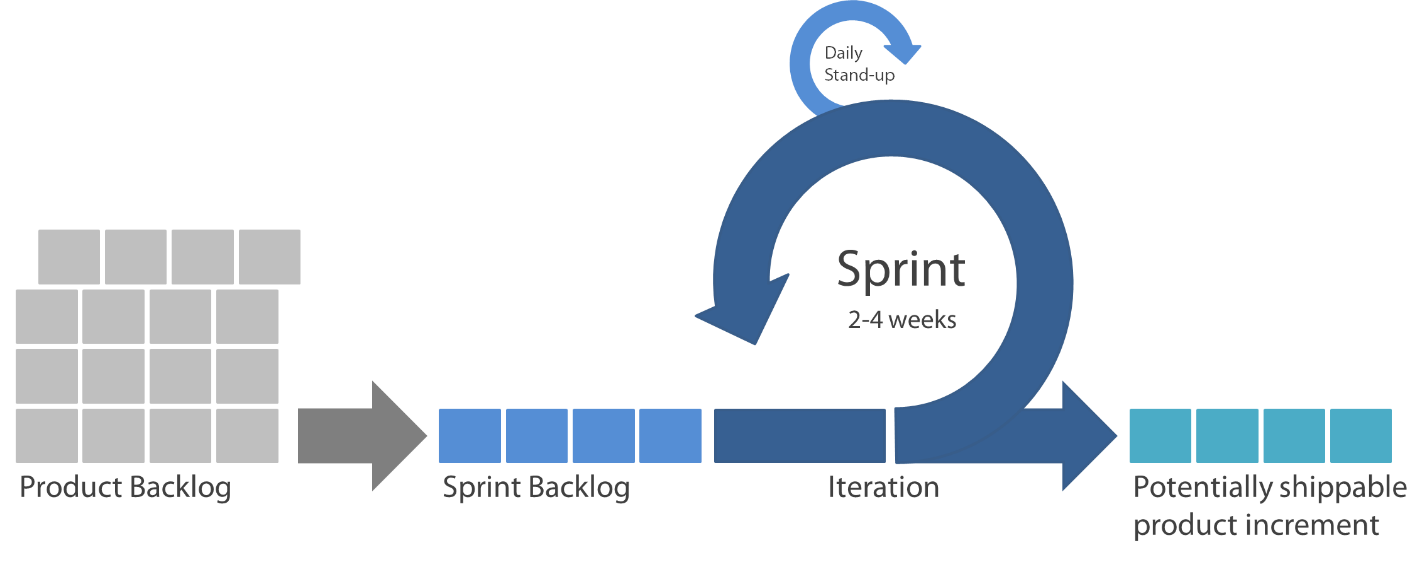


Figure 3 General Agile Scrum Diagram (Bogazici\_University, 2016)

Reviews during the collaboration phase ensure that sprint development is kept on track. Once collaboration is finished when the sprint concludes it’s testing. The deliverable created, elicits feedback based on the suitability of the feature, and may produce more tasks for the product backlog. This allows a feature to be refined effectively during development rather than having a retrospective only at the end of a period of work. It is instead incorporated into the work loop of the project.

This development methodology is specifically relevant to the project undertaken, as Agile Scrum is intended to allow product design and architecture to emerge as a consequence of sprint completion, rather than a traditional pre-set goal (Schwaber & Beedle, 2002). Specifically, as this project is largely exploratory in nature, the final design and architecture of the project will be able to be built iteratively as the project itself evolves in understanding.

Sprints, the backlog, and stand up meetings are further elaborated upon in section 7 Project Management.

## 4.2 Technologies

A brief introduction to the technologies in use during the project, and the reasons for their usage.

### 4.2.1 Apache/PHP

Apache facilitates the use of the project website front end (The Apache Software Foundation, 2019). This in turn hosts the PHP submission point for the sent data (The PHP Group, 2020).

Specifically, Apache handles the HTTP protocol and it’s connections, while PHP handles any data submission, processing, and response generation for the Collection web service.

Apache itself generates a HTTP response code upon submission of data to the service, and PHP also generates a response within the body of the HTTP response in JSON format. The former refers to the status of the protocol itself, and the latter is specifically for the Gatherer application to be informed of the service’s status regarding any submitted data.

Further, PHP interfaces with the internal MySQL service to make requests for information regarding existing data, and to store new data when needed.

The developer chose these technologies specifically for their established use in the industry, and familiarity with their use.

The usage of this technology is expanded upon in section 5.1.2

### 4.2.2 MySQL / Database Development

MySQL serves as the back-end data storage repository for the pre-analysis data collected by the Gatherer application. The database comprises of several relational tables which efficiently store the sent data. These were developed in conjunction with the Collection phase to store only the needed data.

The use of this technology is expanded upon in section 5.2.1

# 5. Deliverables

This section will explain the development of each deliverable, include explanations of the implementation, any difficulties faced during development and their mitigations.

## 5.1 Collection Phase

This phase is responsible for the principle collection of data from each respondent’s system. This includes the gatherer program, and the front-end of the cloud service.

### 5.1.1 Gatherer

#### 5.1.1.1 Code

The originally planned flow of the program updated on the 15/02 in response to usability testing, and user experience considerations (See Appendix C). During testing, it was discovered that there was an unnoticeable difference in program performance between the original version (switching ConsoleState from the execution environment choice State, enacting tests in a new thread while asking for data sending permission, and then showing results), and the second version of enacting the states in sequence. The second version also enables the user to be informed of program operation during testing, and of the data sent before consent is given. Further the original version’s order of operations, while intended to improve application performance, was not congruent with ethical approval.

A close up of a map

Description automatically generated

Figure 4 Flowchart describing the flow of the program when in use

Further refinement of user interaction resulted in the ConsoleState flow described in figure 4. In this, further functionality for debugging the output of tests, and error presentation functionality was introduced.

The in-program error presentation functionality prevents exceptions from escaping the program and incorporates the error messages from both system generated errors, and developer included user-friendly errors. This is done, with the presumption that users which understand the error will be more likely to resolve the issue and submit their data, and the goal of handling all program exceptions without halting the program.

The debug functionality is included for debugging and not advertised on the initial screen presented to participants. This prevents participants from viewing the data in a non-contextualised instance, which could cause confusion. In the event that a participant does utilise this screen the screen does not forward to the sending stage, which prevents the sending of data without consent being given.

#### 5.1.1.2 Design pattern choices and SOLID adherence

This section references the Gatherer’s UML diagram in Appendix C.

Several design patterns have been incorporated into the design of the Gatherer application. These are implemented to increase maintainability, and better adhere to the SOLID design principles.

The primary pattern implemented is the state pattern in the form of the ConsoleState. The main program loop, located in the Gatherer singleton and executed by the main thread, utilises this ConsoleState to proceed through the programs intended operation order. It does so by changing the state of the current state to the next state, and processing it. This follows the intention of the pattern to change behaviours based on the current state, and drives the core usability of the program.

Secondarily, the Singleton pattern is used to restrict the creation of certain classes to one instance only. The program uses this to ensure that these classes are persistent through the lifetime of the program. This is evident in the main program class, wherein both the DataStorage and Gatherer system are instantiated. These are maintained until the program exits.

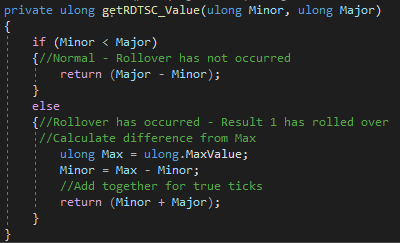
The Adapter pattern is used as inspiration regarding the implementation of the DLLUsage class. This class adapts the required instruction call and measurement methods into calls to either a windows DLL or a Linux shared object, depending on which operating system is executing the program.

#### 5.1.1.2 SOLID Adherence

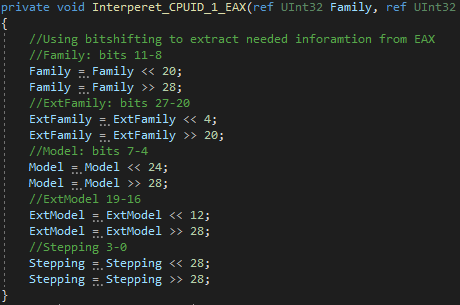
The principles of SOLID have been adhered to where possible. This section briefly discusses the main points of adherence to SOLID principles.

##### Single Responsibility

Summarised: Each piece of code should have one responsibility for which it should carry out.



The code snippet from Gatherer.Testing.Test is responsible for returning the difference between two RDTSC measurements. It does so, and accounts for the case of a time stamp rollover. This follows the principle by only being responsible for a single ask, to return a correct measurement between the two values.



This code snippet from Gatherer.Testing.Extract does not follow this principle. This method has multiple responsibilities as it returns multiple values as references. Similar interpretive methods operate in the same way and do so due to the iterative testing employed during development. These methods should be reviewed and implemented correctly for the eventuality that the suppositions which allow correct operation change.

##### Open/Closed Principle

Summarised: Software is open for extension, but closed for modification. Implementing inheritance or interfaces to enable class substitution, and prevent modification.

This is employed in multiple respects in the program, most notably the ConsoleState inheritance, and the SharedLibrary inheritance.

However, if the original layout of the program persisted with a separate WindowsDLL and LinuxSO class this principle would have been violated. As the classes would no longer have been able to substitute for each other within the program (See Appendix C Gatherer Final UML).

#### 5.1.1.3 Implementation of Brengel et al Methodologies

Brengel et al describes three methods to detect virtualised execution. The first two rely upon RDTSC instructions to measure their time taken in CPU cycles. The introduces a novel method. The methods and their implementation are described below.

##### Method One: CPUID

The first method is the simplest to incorporate into the Gatherer program, its functionality relies solely upon the DLL procedures and requires minimal interpretation of the RDTSC instruction outputs. Method one leverages the dissimilarity between CPUID and NOP instruction measurements identified by Brengel et al.

A screenshot of a cell phone

Description automatically generated

Figure 5 Hardware assisted virtualisation diagram (Murray, 2019)

This occurs due to the hypervisors trapping special/secure instructions in a VM-Exit, such as with the CPUID instruction (Intel, 2019), while non-privileged instructions are executed normally (Figure 5, Figure 6). The disparity between virtualised and non-virtualised instruction timings make this easily discernible (See section 6.2.1 Preliminary Results).

A close up of text on a white background

Description automatically generated

Figure 6 Simplified diagram of execution within a virtual machine

Brengel et al condenses the result of this method into a NOP/CPUID ratio, while the Gatherer retains the distribution of the results (Minima, Lower Quartile, Median, Upper Quartile, Maxima).

##### Method Two: TLB Eviction

Method two takes the longest timeframe to complete. On the development system both method one and method two return results after one and a half minutes.

Method two was intended to test 64, 128, and 256 pages of memory for accession times. This was to occur using the same procedure as Brengel et al with the aforementioned pages being, initially written to and read from to establish TLB entries, then measured accessions to the pages to establish the base TLB measurement, and then a final round of measured accession with a CPUID instruction called before each accession.

A close up of a piece of paper

Description automatically generated

Figure 7 Simplified representation of a TLB-Miss and TLB-Hit

The hypothesis was that the VM-Exit caused by the CPUID instruction would replace the TLB entries previously established due to the external processing required by the hypervisor, causing TLB-Misses that would be discernible from normal accession measurements as seen in figure 7.

However, the implementation was met with a problem in the way that the TLB eviction procedure needed to function. Due to the replacement of general register values by the CPUID instruction, references to the memory page would need to be pushed onto the memory stack before CPUID was called and popped back into the register after the call. However, this caused several memory management errors during testing which were unable to be resolved. The result is that this method implements a safe version of TLB eviction, which likely will not yield any results. This is further discussed in section (5.1.1.4 Dynamic Link Library).

##### Method Three: Stealth CPUID

Method three takes the base understanding from method one, that the CPUID instruction completes slower under virtualised environments, and uses this as a pseudo timing source. This also changes the metric used, with method one the metric can be said to be proportional to the affect virtualisation has upon the measurement, rather method three is inversely proportional. The third method provides that when time constrained, native CPUID calls can occur more frequently, and virtualised CPUID calls occur less frequently.

A screenshot of a cell phone

Description automatically generated

Figure 8 Non-representative visualisation of Virtual and Native CPUID call times

This method is enacted within the gather program using the DLL’s blank CPUID procedure and internal loops. As described in Brengel et al, there are two loops. One increments a counter after a CPUID call, and the other is a busy-wait loop to establish an outside limit to loop execution.

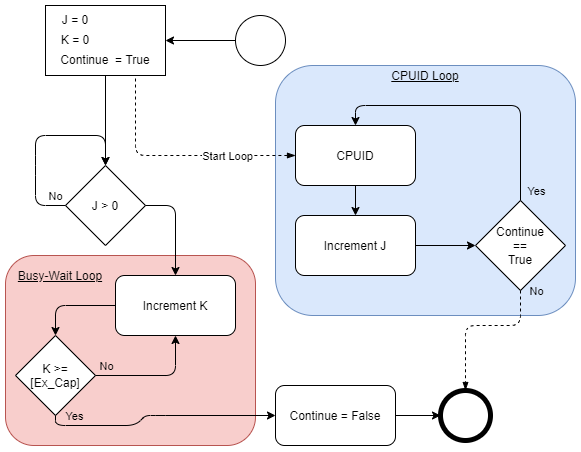


Figure 9 Visual representation of the Method Three implementation

As seen in figure 9, this method has been recreated to be near exact to the method described. The differences noted, are the inclusion of an “Ex\_Cap” variable in the Busy-wait loop, and a “Continue == True” decision in the CPUID loop. This Ex\_Cap sets a limit on execution which was not readily established in the primary research. The decision in the CPUID\_Loop provides the thread responsible for the loop an exit condition. Testing found that an Ex\_Cap value of 800 was suitable to recreate the same general threshold as the primary research during testing.

Differences to the visual representation include the use of a shared “CPUID\_Iterable” object to store both the “J” and “Continue” variables. This allows the both the busy-wait loop, and CPUID loop to make use of the lock functionality to inspect the shared object, making the method thread safe.

#### 5.1.1.4 Dynamic Link Library (DLL)

To prevent the potential for the C# compiler to interfere with measurements, the decision was made to utilise a Dynamic Link Library (DLL) for specific instructions. Written in assembly, the DLL is more directly controlled than general C# code, the execution of instructions occurs in the order it is written without being beholden to any code optimisations inherent in general language compilers. Section 5.1.3 further expands upon the DLL.

In testing the DLL procedure “MeasureMemoryCPUID” was unable to correctly execute. During debugging calls to the procedure are followed by a protected memory exception which prompts the Error ConsoleState. However, before the application closes it can be seen that the procedure does return the measurement of the memory accession. Debug testing showed that the “MeasureMemory” procedure, which does not include the CPUID call, returned 137 cycles. The same number of cycles as the previous procedure inclusive of CPUID call on the native debugging system.

To rectify this the initial “MeasureMemory” procedure was called after prompting CPUID 0. Brengel at al states that “a notable difference in access times when revisiting the evicted page” can be measured. But this did not occur. On both the native, and virtualised debugging system measured cycles were equal. As such these procedures are not fit for purpose, and will need to be revised in a future implementation.

#### 5.1.1.5 Data Sending

Data sending in this application comprises several parts, encoding and shuffling, encryption, and sending.

The data sent is first encoded in the format described in Appendix C Gatherer Encoding, in the general format Code, Length, Values. This enables each sent portion of data to be easily identifiable by any program/service which shares this format. And plays an important role in error detection.

The data was planned to then be encrypted using RSA for submission to the cloud service. This included the generation of a keypair which can be found in the root of the code submission directory. However, during development it was too difficult and time consuming to align both the Gatherer and Cloud collection service in this regard. This effort was intended to start the process of generating the public key in an obfuscated manner but was unable to reach this point due to the aforementioned problem. This is further discussed in section 5.1.1.6

Sending the data is facilitated by the form submission features of the HTTPClient. This client interacts with the PHP form submission service in the AWS collection deliverable. This is achieved through a multi-part form submission, and a reading of the response in JSON format. This response is displayed to the participant in the event of an error, but ignored upon a success response in favour of the “ThankYou” screen.

This form submission did encounter issues with acceptance by the PHP submission point, but was resolved through the usage of the multi-part form submission, rather than a standard form submission. This incorrect submission can be seen in packet four of the “Submission\_Incorrect.pcapng” file and seen to be corrected in packet four of the “Submission\_Correct.pcapng” file in the project directory “site\_debug”.

While the final deliverable is able to send data to the internet connected server correctly, it does not conform to the requirements promised as part of the ethical approval process.

The ethical approval document submitted for project approval (See Appendix I) specifically states that submissions will be encrypted. This renders the approval to collect information from systems and participants over the internet void. This is partially mitigated through the collection of data from systems in a private capacity which enables the project to continue.

#### 5.1.1.6 Feature difficulties

Certain features of the application were partially implemented:

##### Method Two

Method two was deemed to take an unacceptably long amount of time for a user to allow the program to operate without cancelling the tests. Further, the changes to TLB eviction measurement to enable the return of results may have resulted in the inability of any measurable difference to be detected. This method has been kept in the program, but only tests a block of memory 64 pages in size, and is not expected to yield any results.

Certain features were unable to be implemented:

##### Microcode Revision extraction

Microcode Revision collection through the DLL conflicted with the core requirement of not requiring elevated permissions. This initial method of collection required either an operating system executing code in Real Mode, or access to execution at the kernel privilege level in protected mode (Figure 10). Collection of this piece of information is not be possible in this iteration of the application as the information sources were deemed to be alterable. In a future iteration of this application that does not rely upon the utmost accuracy of the reported data it could be sourced in both Windows and Linux. Under Windows a non-privileged registry query retrieves the information, and under Linux this is available in /proc/cpuinfo

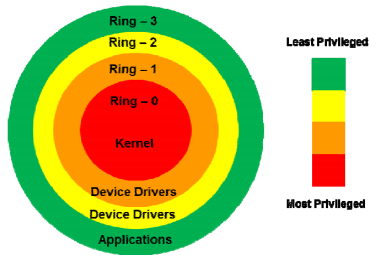


Figure 10 Protection rings in x86 CPU architecture (Yeun & Zemerly, 2012)

##### Multi-platform support

While Dot Net core was chosen for it’s multi-platform support with the intention of supporting at least windows and Linux, development was unable to translate the assembly DLL into a Linux shared object library for use on the platform. This has meant that the only data sources are those running the windows operating system. This has occurred due to time constraints, and is discussed further in the project management section. The program is able to start under both platforms, but Linux execution is unable to load the Windows DLL.

##### Unable to encrypt sent data

Efforts to encrypt the data before sending failed. During development either the Gatherer itself or the Collection service were unable to encrypt or decrypt the data. Due to time constraints it was deemed to not be a primary concern of the project to encrypt the data, as enough data could be collected personally to achieve a minimum viable product whilst maintaining the achievability of most project aims.

This feature has been removed, and replaced with Base64 Encoding the data. This does not imbue the sent data with any security, and is not intended to be used by the general public.

Further a form of white-box cryptography was intended to be used to generate the public key in the encryption, providing a layer of obfuscation to any reverse engineering attempts (Wyseur, 2012). By the intended design this generated key would then be required to send any incorrect data to the collection service, thereby frustrating reverse engineering attempts beyond the lifespan of the project. This feature was intended to constitute a deterrence by denial of benefit as described by (Jasper, 2015).

### 5.1.2 Amazon Web Services Cloud front-end



Figure 11 Brief overview of cloud infrastructure virtues. Partially adapted from (Pantic & Babar, 2012) and (Tomas & Tordsson, 2013)

The plans for the AWS solution included scaling in order to process the expected demand on system resources. The front-end would be responsible for the Apache web service, file hosting, and initial data submission/processing through PHP. This would utilise significant resources from a single server instance.

The processing by the initial PHP submission point includes error detection. This is facilitated by the in-built features of the sent data mentioned in section 5.1.1.5

Each submission is checked for a correct code/length setting, decoded into a generic object, and checked for the correct header. An error is returned if any length is incorrect, the decoding runs over the array length, decoding finds an incorrect code, any data point is not set, or there is a MySQLi error. This prevents incorrect data from being processed by the service.

#### 5.1.2.1 Deliverable Changes

Originally intended to be hosted in the Amazon Web Services cloud, this service has been heavily impacted by the circumstances surrounding the novel coronavirus. Funding initially available for the AWS cloud infrastructure needed to be diverted, and a migration to an on-premises solution was therefore required (See Appendix E).

This on-premises solution does not provide the advantages of cloud infrastructure and incurs significant disadvantages due the age of the hardware. The system N3/V3 referred to in section 6.2.1 is utilised in this on-premises solution to host the required services. Due to this change, the service will be unable to scale.

Further, this solution prevents the optional requirement of “Implement auto-scaling and load balanced HTTP solution” from being enacted, and compromises the secondary objective “Allow for public submissions”. While public submissions will be still be technically possible, the funds for the project’s advertisement to the widest possible public audience are no longer available.

As the AWS infrastructure could no longer be utilised for the LAMP stack deliverable it was deemed prudent to leverage the AWS cloud differently. Rather than hosting the LAMP stack, the funds already invested in AWS would be used to instantiate windows systems for data gathering.

A minor required change to the LAMP stack deliverable, is the change to a windows based personally deployed physical system. This minorly changes the LAMP (Linux, Apache, MySQL, PHP) stack to a WAMP stack (Windows, Apache, MySQL, PHP).

## 5.2 Analysis Phase

### 5.2.1 MySQL

The primary storage location for submitted data, the MySQL database is designed to store these as efficiently as able. The database’s required datapoints were brought through the normalisation process to deliver 5 distinct tables in third normal form.

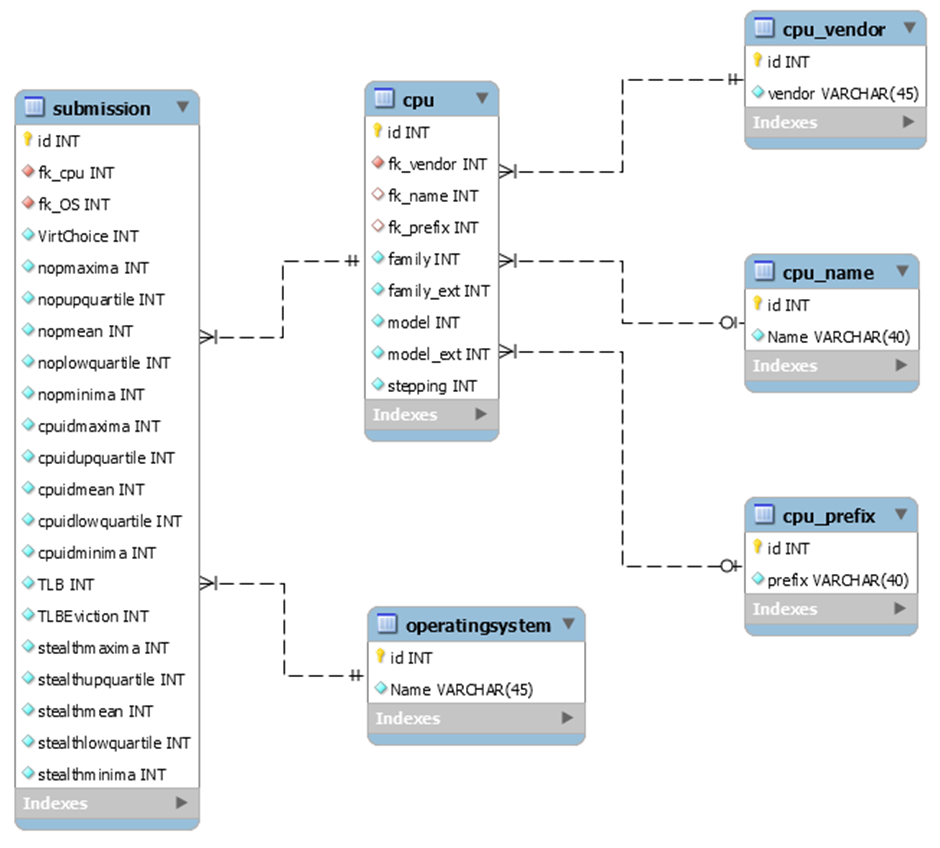


Figure 12 Final EER diagram for the database

The finalised entity relationships of the database reflect the initial intentions for the project. Naming capability is included, as the initial thought included scraping the CPU name and prefix from the OS to give context to the other stored CPU attributes (See Appendix F).

This, along with the CPU capability flags, was not implemented in the final deliverable. The former is due to time constraints, however the latter was deemed to be inconsequential to the storage of the results. The final result is shown in figure 12.

A screenshot of a cell phone

Description automatically generated

Figure 13 User account write access to the database

Considering the potential for MySQL injection from the PHP submission point, the database was segregated regarding permissions. The existing root account was used to create two new accounts, jmurray, and HAVD.

jmurray was created specifically to be used by the developer to debug any submission uses from the PHP submission point, and as such has full read, write, and delete permissions on the database. This does not extend to GRANT permissions to avoid the potential for complete database compromise.

HAVD has very limited permissions and is used by the PHP submission point. It has SELECT permissions to all tables in the submissions database, and can only write new records into cpu and submission. As the HAVD account is responsible for executing any sent details from the Gatherer program, this prevents any potential SQL injection from deleting or modifying any part of the established database design.

This deliverable provides the needed storage capabilities, and does so with support for the PHP to securely access the database with the HAVD account.

#### 5.2.1.1 Feature Difficulties

While the primary function of the tables was to store the submitted data for analysis, they also provide a secondary capability to be manually named. However, the other areas of the project were deemed priorities for development. As such this additional feature was not given any weight as a backlogged product, and did not receive any more time beyond the initial inclusion into the database’s structure.

If the intended execution of the project had occurred, this would have been used to label processors in the analysis stage of the project and given context to CPU comparisons inter-generationally and intra-generationally.

### 5.2.2 R-Studio

This deliverable focuses on the analysis and methodology used to produce indicative visualisations.

#### 5.2.2.1 Hypotheses

1. Instructions processed by the hypervisor may have a measurable latency increase

Post reading of Brengel et al:

* 1. If the hypervisor affects the latency of CPUID calls, there should be noticeable bands created by native/virtual results when graphed as CPUID/NOP.
  2. If a program has cached memory address translations in the TLB and initiates a VM-Exit processed by the hypervisor, the TLB may lose it’s cached translations.
  3. If Virtual machines are more negatively impacted in CPUID calls, it follows that a system experiencing high resource utilisation will experience a higher negative latency impact.
  4. If resource utilisation increases measured CPUID latency, and if this is due to context switches, it follows that the virtualised results of method 3 may be affected under high resource utilisation.

#### 5.2.2.3 Hypotheses explained

As this is an exploratory project, the methodology used is not fixed and instead guided by the pre-existing knowledge of the studied technologies.

1. This hypothesis is indicative of the base understanding of hypervisor functionality. As it is a layer between the underlying physical hardware and the operating system supporting an application, base latency should be above that of a native system in some cases. This is tested in this project through method one of Brengel et al, and is clearly visible in the latency of CPUID instructions as the hypervisor specifically traps these (Intel, 2019). As such, this should be clearly seen when attempting to replicate the results.
   1. These bands should be a result of the relationship between CPUID and NOP instructions. Native systems can be seen in Brengel et al, to not correlate to any noticeable degree, but virtualised systems seem to have a linear or exponential relationship. NOP latency may correlate with a larger increase in CPUID latency, causing the banding in visualisations.
   2. This hypothesis is not definitive, as there may be a negligible difference in measurements as noted in Brengel et al, requiring a difference in methodology for any detection to occur as stated.
   3. Further testing is needed to establish whether this relationship exists. The potential exponential CPUID/NOP relationship begs the question as to whether the physical host’s CPU load can affect noticeable change in the VM. To do artificially creating usage may induce this response.
   4. Related to the previous hypothesis, the use of multiple systems on a single physical system produces context switches which “can significantly interfere with the execution of the tasks” (Phan, et al., 2013). It follows that due to the large number of loops, more context switches will take place. If more context switches occur during the busy wait period, the iterative CPUID loop will have longer to run a consequence. This may result in measurement instability.

As this is exploratory work, the majority of analysis occurs at the point of result generation. For discussion regarding the analysis and insights derived from this process see section 6.2.

## 5.3 Implementation Phase

### 5.3.1 Neural Network

The neural network deliverable was intended to accept the COUID, NOP, TLB, TLB eviction and Stealth CPUID measurements to output a classification of True or False for virtualisation detection.

This was to be trained upon the findings of the analysis deliverable. Wherein the structure of data would inform the areas of data points which constituted virtualised or native execution. This is discussed in section 6.2 Analysis.

Unfortunately, due to time constraints this deliverable was unable to be produced, and did not leave the design stage after analysis had occurred. A windows GUI application intended to demonstrate the NN’s functionality was planned, but only the base frame was produced to later include the NN. This can be found in the code directory root in the “NN\_GUI” directory.

#### 5.3.1.1 Feature Difficulties

As the Gatherer application was unable to correctly implement the TLB eviction method, these inputs were unable to be included in the plan for training.

Further, due to the time constraints only a base collating of virtual/native CPUID, NOP, and Stealth CPUID measurements was graphically collected to begin the process of Neural Network training.

# 6. Project Results

This section evaluates the final deliverables against the core, secondary, and optional requirements in section 3.1

However, the analysis phase will discuss the results instead.

## 6.1 Collection Phase

The collection phase implements all core requirements, allowing the project to extend to through to the analysis phase by providing the foundation of the project.

While minimally viable, this phase does not fulfil the secondary requirements. Microcode was unable to be collected. Linux was unsupported. Public submissions were severely impacted, to the extent that this requirement is not deemed to be met. The secondary method of Brengel et al was also unable to be correctly implemented. HTTP balancing was also unachievable due to the diversion of funds. This has meant that extensive data collection wanted, was not possible. Further objectives have been impacted by this.

The project did follows SOLID principles, but this was secondary to the main project objectives. Ultimately the collection phase only meets minimum viability.

## 6.2 Analysis Phase

### 6.2.1 Preliminary Results

Preliminary data from method one suggests that the novel researches results are accurate.

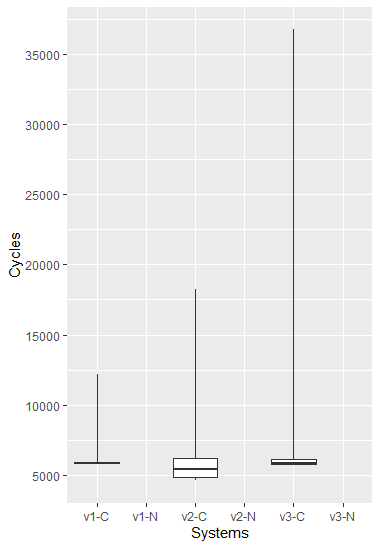
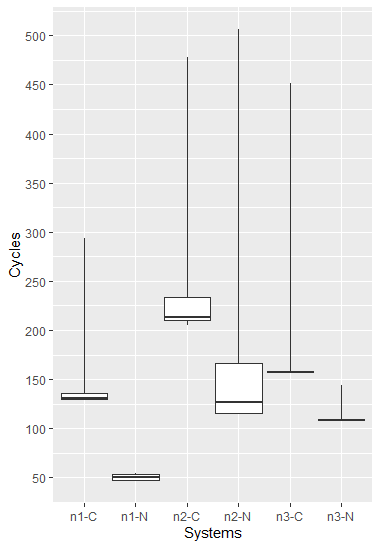
These preliminary results are collected using the debug functionality of the gatherer program. They show a clear delineation between native and virtualised execution, exploitable by the final NN deliverable.

Figure 14 Boxplot comparison of Native(n) and Virtualised(v) CPUID(-C) and NOP(-N) instructions on 3 systems(1-3)

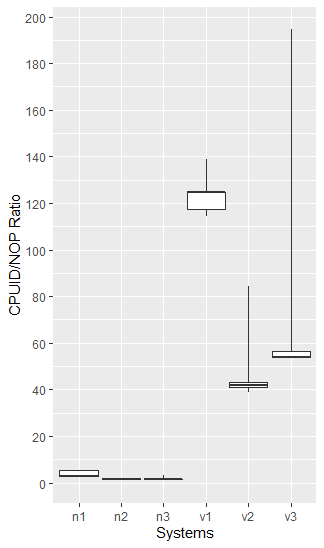
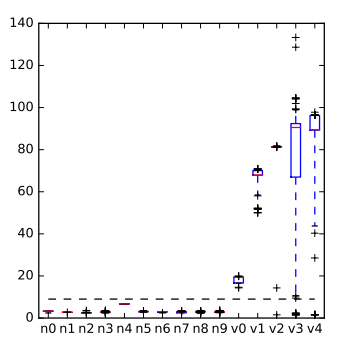
The negligible difference between virtualised and native NOP execution is intuited to be due to differing levels of idle system utilisation. As this would affect the number of instances of context switching between measurements.

Figure 15 Method One CPUID/NOP Ratios (Brengel, et al., 2016)

Brengel et al, shows that the virtualised systems’ (v1 – v4) CPUID to NOP ratio trends beyond that of native systems (n0 – n9). Similarly, the preliminary results show the same

trend with native systems’ maxima ratios (5.3 – 0.9) being significantly lower than their virtualised counterparts (194.5 - 84.3).

The base CPUID/NOP ratio discrepancy between preliminary results and that of Brengel et al, can be explained by the gathering of results from systems operating under windows 10, rather than a Linux operating system. This would not be merely explained by programming differences, as the Gatherer utilises an external DLL written in assembly to separate any potential compilation interference from the C# dotnet core’s JIT (Just In Time) compiler.

Figure 16 Method One CPUID/NOP Ratios, Preliminary results

However, the preliminary results are compatible with that of Brengel et al, as both sets of data comport with the described ratio threshold of 9 (Brengel, et al., 2016).

### 6.2.2 TLB Evictions

TLB Evictions will not be analysed due to the problems discussed in section 5.1.1.6 Method Two.

### 6.2.3 CPUID and Results affected by heavy CPU workload

During observation system 3 was noted to have certain inconsistencies in results with method one, and three. This was hypothesised to be due to the age of the system, as it utilises a circa 2010 triple-core AMD phenom processor, and the workload expected of it.

As such system utilisation was deemed to be an area of concern regarding the potential validity of Brengel et al’s results within host systems with a higher number of virtualised tenants. This would be especially concerning in virtualisation environments where overbooking is used to increase the efficient utilisation of host system resources (Tomas & Tordsson, 2013), and could lead to inaccurate results. This inaccuracy could be worsened as a result of context switches between the two measurements (Phan, et al., 2013). Method three in particular could be vulnerable to these lengthening times in both the CPUID loop, and the main busy wait loop.

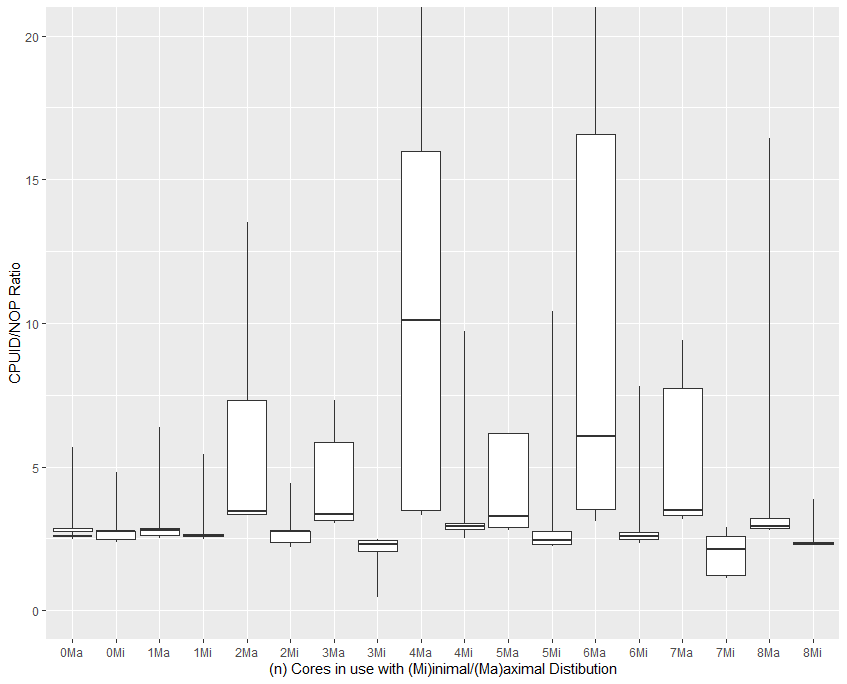


Figure 17 Zoomed distribution of native CPUID/NOP ratio's under differing system load

These results have been collected by artificially increasing the workload of the host systems CPU. This was achieved by running a differing number of workloads in Prime95 under the “Small FFT’s (tests L1/L2/L3 caches, maximum power/heat/CPU stress)” setting. The “(n) Cores in use” referenced in the visualisations, refer to relative utilisation of the CPU verified to be in use during testing. With 4 at 100% of the physical capacity of the CPU, and 8 at 100% of the logical capacity.

Figure 18 shows the results of testing undertaken from a native system to test the affect of host system CPU utilisation on the CPUID/NOP ratio. These distributions show the potential extremes which the ratio may be, and have been chosen to more easily visualise the possible ratio’s under these conditions.

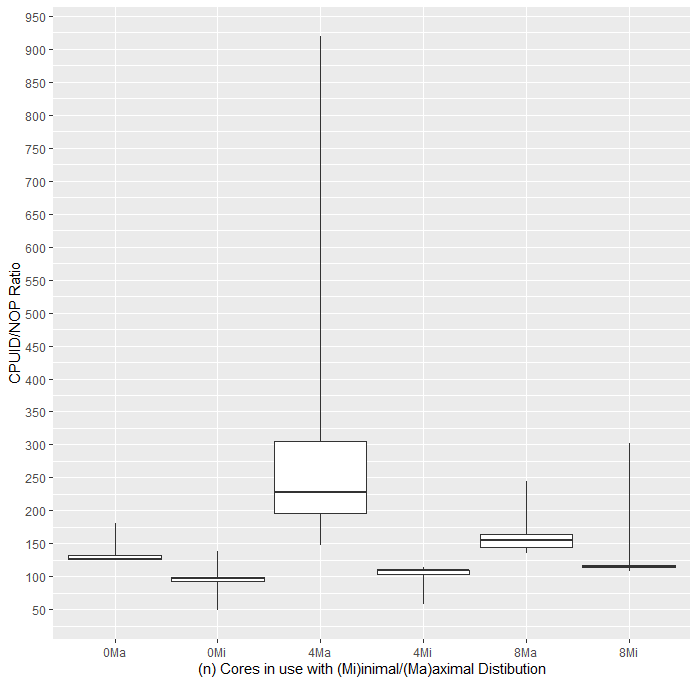


Figure 18 Distribution of Virtualised CPUID/NOP ratio's under differing host system load

As can be seen in the figure, it is entirely possible for a native system under load to breach the threshold set by Brengel et al. However, this data also shows that while the system load increases the minimal distribution of values is well within the threshold. This points to the viability of distinguishing Native/Virtual performance through the observation of CPUID/NOP minima.

But this would not be based on the performance characteristics of the Gather program and DLL created, which could differ from Brengel et al due to the mixed usage of an Assembly DLL within a C# core program. In testing, the highest native ratio was observed at 23.06, and the lowest virtualised ratio was observed at 48.65. Taking into account the characteristics of the virtualised instance when the host system is under load (figure 15), virtualised execution is still very distinguishable from native when using the CPUID/NOP ratios received from the Collection phase.

A screenshot of a cell phone

Description automatically generated

Figure 19 Stealth CPUID calls under differing host system utilisation

A close up of a building

Description automatically generated

Figure 20 Zoomed Stealth CPUID calls under differing host system utilisation

The final method to evaluate from Brengel et al, is the CPUID Stealth method. This method aims to limit CPUID calls in virtualised environments to limit the possibility of detection by dynamic analysis.

As shown in figure 21, CPUID calls are confirmed to be limited even under system stress. However, under a virtualised environment the number of CPUID calls can increase above the Brengel et al threshold of 4. The data relating to this phenomenon shows that there is a remarkable increase in outliers when the host-system is more heavily utilised (Figure 22). The results obtained from the Gatherer program clearly show that the threshold of 4 set in Brengel et al is not suited to virtualisation detection in all circumstances.

These characteristics make creating a strategy for Neural Network training challenging.

A close up of a white wall

Description automatically generated

Figure 21 Distribution of Stealth CPUID calls under load in a virtual environment

### 6.2.4 Conclusions

#### 6.2.4.1 CPUID/NOP Ratio

For the ratio of method one, it is fit for purpose. Due to the self-tying nature of the ratio, the ratio remains accurate at higher system utilisations. Through testing it is determined that the upper ratio for native execution is 23, and the lower ratio for virtualised execution is 48.

As such, the limit for native/virtual determination will be set at 25 for Neural Network training.

#### 6.2.4.2 Stealth CPUID

While this method of looping to determine virtualised execution does operate accurately under the majority of data points, it operates less-accurately at higher host system utilisations. This threshold will be kept at 4, as set by Brengel et al.

However, to mitigate the inaccuracy issue rather than training on a set threshold the Neural Network will be trained to prefer the CPUID/NOP ratio unless the stealth method indicates virtualised execution. If not able to do so, the program implementing the NN will prefer the Stealth CPUID indication over the NN’s output.

## 6.3 Implementation Phase

While great efforts have been made to mitigate the unexpected consequences of the disruption faced, this phase was unable to be completed.

This phase did not leave the design stage, this phase did not complete, and did not meet any of the requirements set out.

While the analysis phase sets out goals which are eminently achievable through neural network creation and training, these were unable to be implemented.

Specifically, through analysis it seemed feasible for the CPUID/NOP ratio measurement to have an effective True Positive rate nearer 100%, and False Negative rate nearer 0%. Improving upon Brengel et al.

# 7. Project Management

Following agile meant that deliverables were completed in sprints. The planned sprints for the project can be seen in the timeline plan in Appendix B. These planned sprints constitute the start of the project including all collection deliverables, and the start of the analysis phase in the form of hypothesis creation.

However, it can also be noted that these sprints were not kept up to date, and subsequently the project did not have a clear path to completion. This was compounded by the lack of updates to the planned timeline.

This mismanagement resulted in the

# 8. Project Post-Mortem

While the primary objectives of the project were completed, not all deliverables have been met, and several avoidable feature removals have occurred.

Three of the four primary project objectives have been successful despite these mitigations.

Brengel et al has been verified in its main assertions as a viable foundation for virtualisation detection, and avenues for further research and analysis have been correctly identified.

The data collection framework is functional, but does not provide the needed protections for public use as data is not encrypted in transit and only marginally obfuscated. This means that currently this deliverable is only fit for private collection, rendering the pool of data to be collected much more expensive to reach in terms of expendable project resources.

Brengel et al, has been evaluated to the extent able and several avenues for continuation have been identified. Further, analysis of these reveal a strong indication of success in any potential implementations utilising this knowledge.

## Implementation vs Minimum Viable Product (overview)

Barring the final Implementation deliverable, this project meets the minimum viable products in all other phases.

## Successes

Analysis of collected data clearly shows the phenomena in action. And further analysis reveals that there is a need for more in-depth research regarding the erratic behaviour in method three to determine its factors as a measurement.

The mitigation of unplanned events was completed successfully. And did not force a project ending scenario. Corrective action was taken to ensure the continuation of the project, and most deliverables were created as a result.

## Challenges

The implementation of Brengel et al’s methods involved careful reading of the original paper. This was not a simple task, and errors were corrected following initial readings. These misunderstandings may be the cause for methods two’s inconclusive implementation.

Due to the unexpected global pandemic, heavy disruption occurred to the project in the form of funding withdrawal, and the need for the sole developer to change address.

This meant that funding for the cloud infrastructure was made unavailable and required a migration to an on-premesis solution. This can be seen in Appendix E.

This migration was successful and hosts the project in a limited capacity allowing the project to continue.

This section will not mention the personal stresses faced.

## Future improvements

Inclusion of microcode from Windows and Linux can be implemented using the mitigations discussed in 1.1.5, and are recommended as a future addition to the program.

Future funding availability must be guaranteed to be available, despite any extreme potential unforeseen events.

It would be prudent to examine the affects of the MFENCE instruction more closely in relation to the DLL to prevent memory exceptions, and potentially accurately measure method two of the primary research.

# 9. Conclusion

The completed deliverables allow a clear foundation upon which to complete the project in short time.

All intended identifiable characteristics from Brengel et al are available for exploitation, and have been given a path to do so.

While not all deliverables have been made it cannot be said that the project was completed. However, the deliverables that have been completed carry out their functions as intended proving a clear path forward for the implementation phase to be carried out.

# 10. Future research

While analysis has shown that the Stealth CPUID method adequately detects virtualisation without a timing source. Further research into reliable timing sources is needed to complement existing solutions as they become mitigated against.

Potential mitigations against this implementation should also be researched. Specifically, the mitigation against CPUID/NOP ratio use through timestamp pausing, specified by Brengel et al, could be worsened by nullifying the Stealth CPUID method. The instability observed, if worsened, could render the method ineffective.

The effects of full virtualisation, pare-virtualisation, and containerisation on the methodologies should also be tested in future to ensure compatibility with these technologies.

The effect of retraining the neural network training to prevent heuristic/hash detection of the solution dynamic malware analysis evasion is an area of interest due to it’s plausibility.

Most importantly however, a much larger dataset of results is needed in order to determine the concluded measurement characteristics more accurately. And further study should capitalise upon this to counter the small size of this field of study (Ioannidis, 2005).

Bib

Ref

# Appendices

The remainder of this document consists of appendix documents and material.

# Appendix A: User Guide

This was not deemed necessary due to the simplicity of user interaction for the gather application.

# Appendix B: Project Management

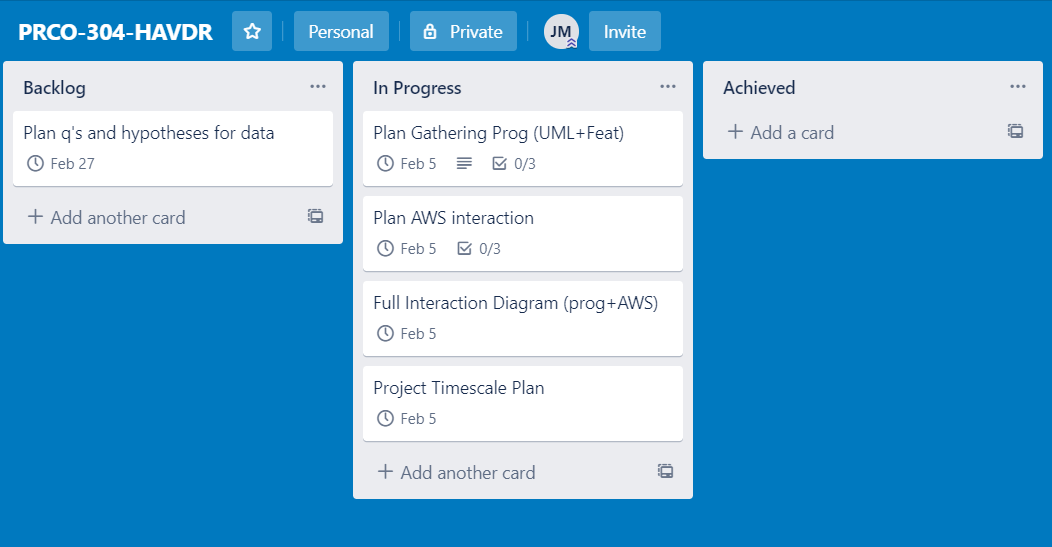
# Annex B Project Management

## Planned Timeline

|  |  |  |
| --- | --- | --- |
| **Description** | **Features/parts to complete** | **Final sprint dates** |
| Initial | * An initial scoping of the project | 07/02/20 (Week 27)1 |
| Planning | * Scoping of components   + Gatherer   + AWS   + Database | 14/02/20 (Week 28)1 |
| Gatherer & DLL  AWS Testing | * Partial Completion of Gatherer * Testing of AWS Implementation | 21/02/20 (Week 29)1 |
| Preliminary Data Processing | * Data questions and hypotheses   + Generate test data   + Graph results using R with test data | 28/02/20 (Week 30)1 |
| Start Deployment | * Start permanent LAMP AWS   + Gather data   + Serve Program | 06/03/20 (Week 31)1 |
|  |  | 13/03/20 - |
|  |  | 20/03/20 (Week 33)2 |
| **DLE Upload** | **Brochure Data** | **25/03/20 (Week 35)** |
|  |  | 27/03/20 - |
|  |  | 03/04/20 (Week 35)2 |
|  |  | 10/04/20 - |
| **End of data collection** |  | **17/04/20 (Week 39)2** |
|  |  | 24/04/20 - |
| **DLE Upload** | **Poster Data** | **27/04/20 (Week 40)** |
| Code & Data review | * Refactor / Review code paths * Review data, graphs, assumptions | 01/05/20 (Week 41)2 |
| **DLE and GitHub** | **Final report and code submission** | **13/05/20 (Week 42)** |
| **Showcase** | **Participation at showcase event** | **14/05/20** |
| **Project Demo** | **VIVA to two supervisors** | **(Week 43)**  **18/05/20 – 22/05/20** |

## Trello Submissions

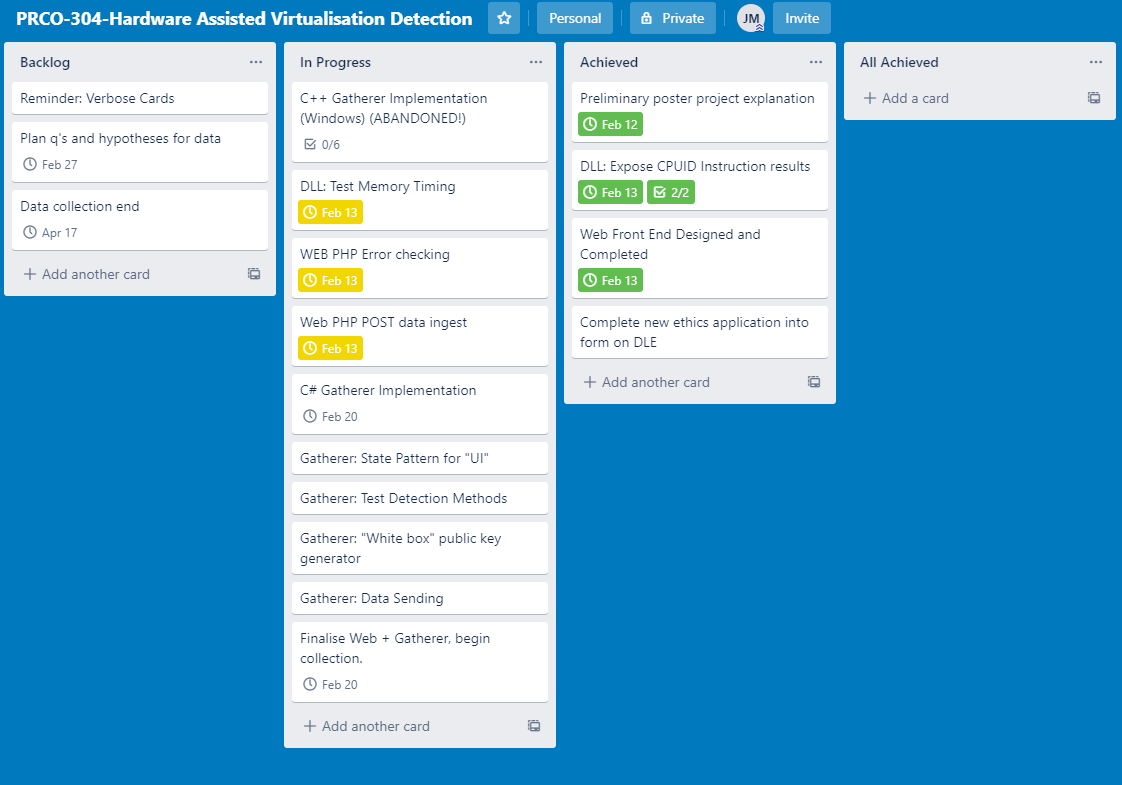
### 01/02/2020



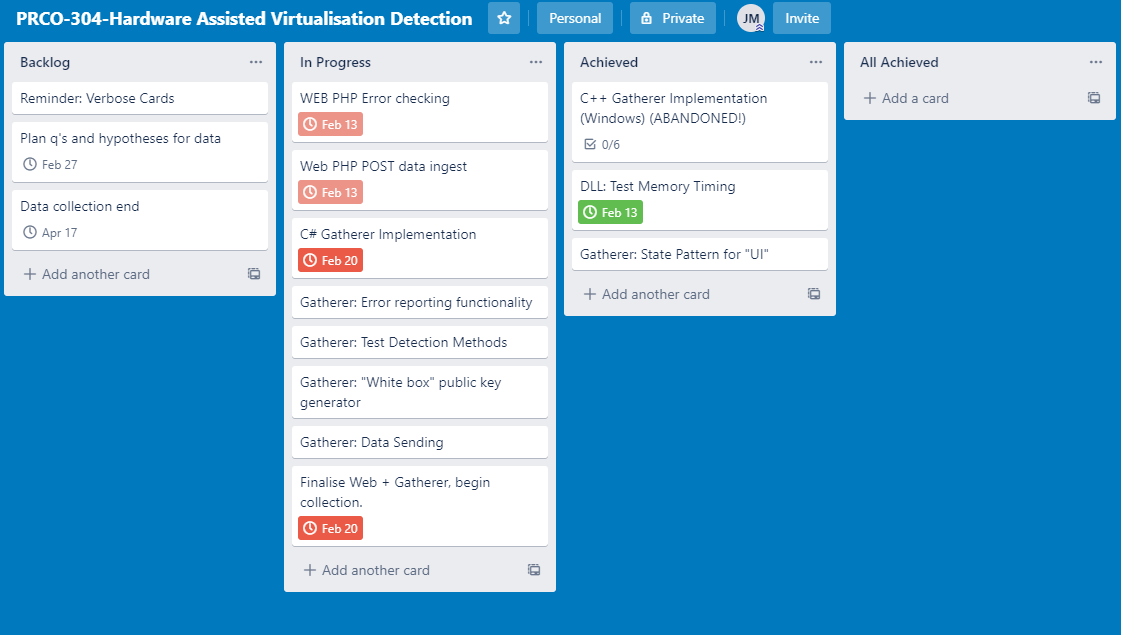
### 06/02/2020

### 

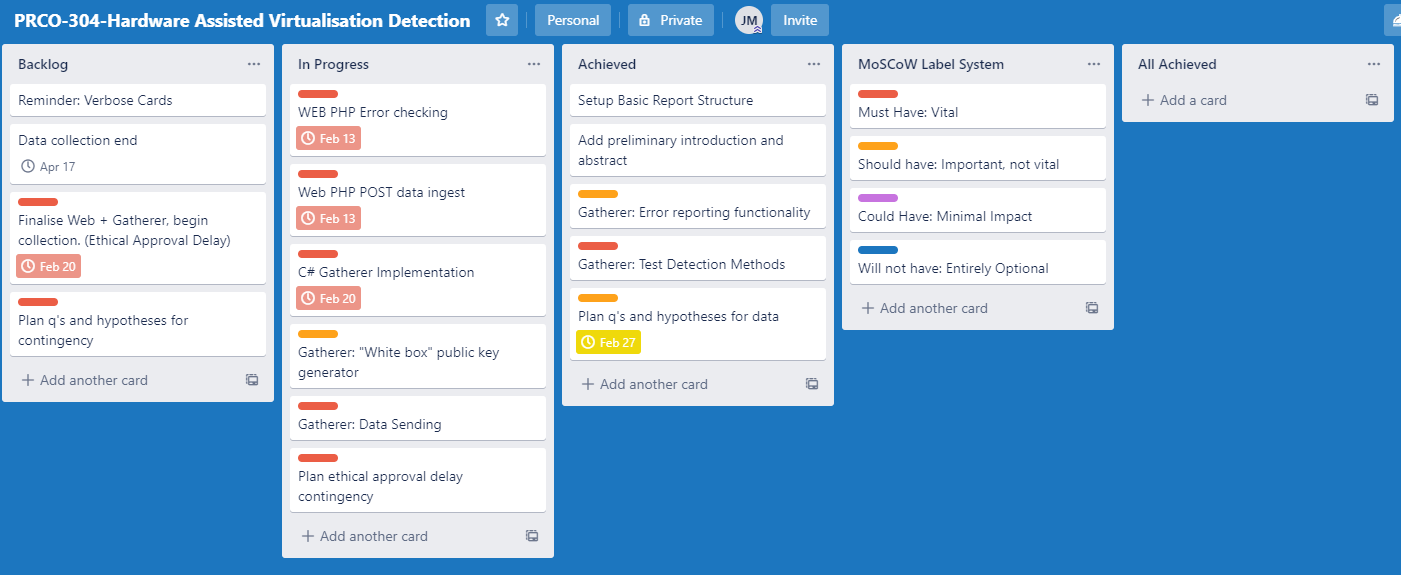
### 13/02/2020



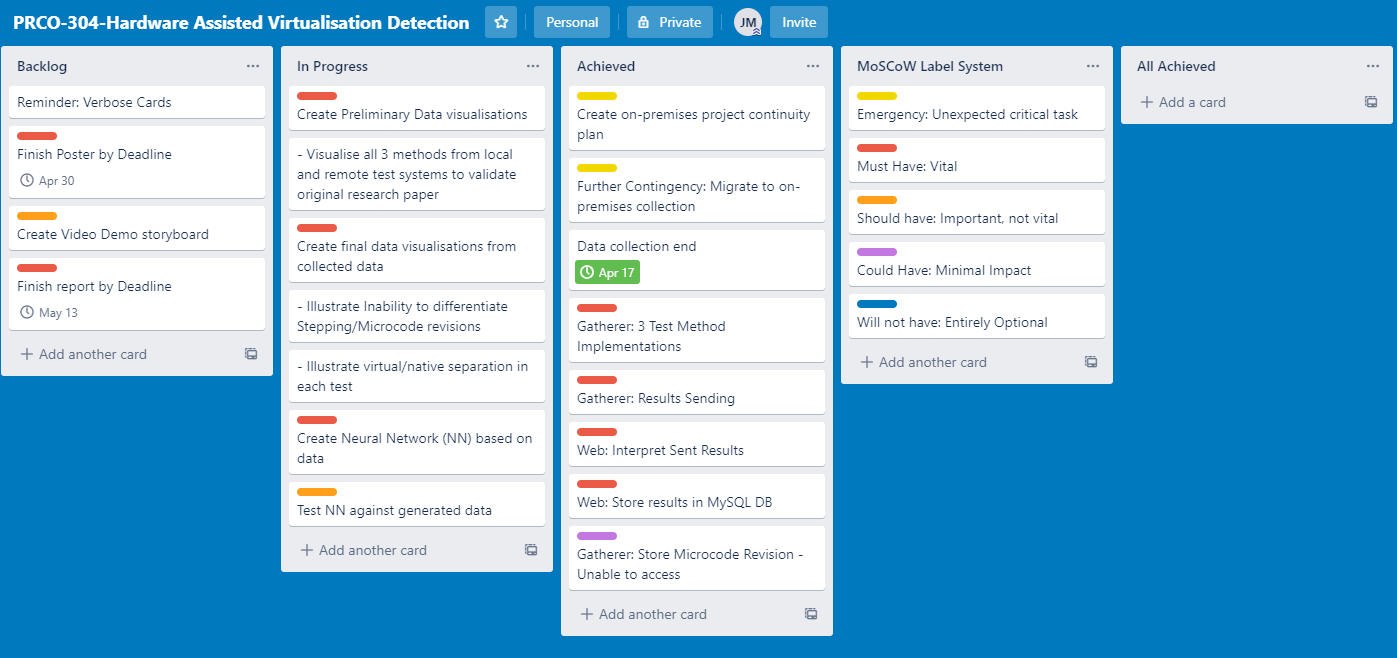
### 21/02/2020



### 27/02/2020



### 24/04/2020



# Annex C Gatherer

## Activity Diagram

A screenshot of a cell phone

Description automatically generated

## Activity Diagram Changes 15/02/20

A close up of a map

Description automatically generated

## Description/Role

The Hardware Assisted Virtualisation Detection (HAVD) Gatherer, gathers the needed information from participating persons. Information from the OS as well as CPUID calls is sent to the Data receiver in anticipation for the Analysis portion of the project.

## Parts

The gatherer consists of 4 parts:

* UI (Command line interface)
  + Displays the interaction with the participant
  + Based on the state pattern, Offers options with which to interact to further states and provide needed information.
* Gathering
  + Responsible for gathering information through the OS, CPUID calls, and detection method tests
  + Utilises a self-programmed assembly DLL made using FASM (Flat Assembler)
* Encoder
  + Packages gathered information for sending with UDP sender
* Sender
  + Sends data to AWS Server

## Prototype Implementations

* Native/Virtual
  + User choice
* AMD/Intel, CPU Model
  + From CPUID Vendor ID
* Method One (CPUID Timing):
  + Calls MethodOneBIAS. This is repeated 5 times
  + Calls MethodOneNOP. This is repeated 5 times
  + Calls MethodOneID. This is repeated 5 times
  + Minima, Lower Quartile, Median, UpperQuartile, Maxima stored in DataGathered
* Method Two (TLB Evictions)
  + Calls
* Method Three (Stealth CPUID Timing)
  + Implemented mostly in C# and partly in assembly.
  + In line with the base paper, the programs main thread calls and starts thread B. Thread B calls CPUID, iterates a counter C and loops. When the main thread detects that counter C is higher than 0, the main thread calls thread B to abort.
  + The resulting counter value is stored in DataGathered.

## Gatherer Encoding

[**CODE**:**Length**:**Values**]

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Code** | **Length (bytes)** | **Values** |
| Header | 10 | 4 | HAVD in bytes |
| User Choice | 22 | 1 | 1-3 (Host, hypervisor, native) |
| OS | 25 | 1 | 1-4(Win, Lin, Free, OSX) |
| Vendor | 38 | 1 | 0/1 (Intel/AMD) |
| Family | 41 | 2 | Family, Family Ext |
| Model | 46 | 2 | Model, Model Ext |
| Stepping | 49 | 1 | Stepping |
| Hypervisor Brand | 52 | 12 | EBX, ECX, EDX (3 x 4 bytes) |
| NOP | 55 | 20 | Minima, Lower Quartile, Median, Upper Quartile, Maxima (5 x 4 bytes) |
| CPUID | 59 | 20 | As above |
| TLB | 63 | 8 | TLB, TLB Eviction (2 x 4 bytes) |
| Stealth CPUID | 66 | 20 | Minima, Lower Quartile, Median, Upper Quartile, Maxima (5 x 4 bytes) |

## Gatherer Final UML

The final UML diagram for the gatherer is overleaf.

## A close up of text on a white background Description automatically generatedFinal Gatherer UML Diagram

# Appendix D: DLL

The DLL implements functions and procedures at an assembly level, for higher level languages to use in more readable implementations.

## HAVD.DLL procedures

* E.g. Procedure Call (Where procedure is used)
* CPUIDVendor (CPU data extraction)
  + Calls CPUID with an EAX of 0H
  + Returns the registers EBX, ECX, and EDX as an Int32[]{1, 2, 3}
  + Used to verify the “GenuineIntel” or “AuthenticAMD” of a CPU independently from OS functionality
* CPUIDVersioning (CPU Data extraction)
  + Calls CPUID with an EAX of 01H
  + Returns the contents of registers EAX-EDX as an Int32[]{4}
  + After CPUID 01H
    - Register EAX contains the family and model information of the CPU
    - Register ECX contains whether Virtual Machine Extensions (VMX) is enabled in the CPU
    - Register EDX contains whether TSC (RDTSC) is enabled in the CPU
* RDTSCTimed (RDTSC Timing Measurement)
  + Calls rdtsc > pop > push > rdtsc
  + Returns the first and second reading of the timestamp counter as a long[]{1, 2}
  + This information allows for the approximation of bias that the measurement process inherently contains
* CPUIDTimed (CPUID Timing Measurement)
  + Calls CPUID with an EAX of 0H
  + Returns cycles taken to execute CPUID
* NOPTimed (NOP Timing Measurement)
  + Calls rdtsc > pop > NOP > push > rdtsc
  + Returns the first and second reading of the timestamp counter as a long[]{1, 2}
  + This measures the CPU cycles elapsed through the use of NOP
  + Used a baseline measurement of the CPU

## Prototype: Testing and Changelog

Versions will iterate until v1.00 which indicates a release candidate.

|  |  |  |
| --- | --- | --- |
| Date | Pseudo Version | Changes Made (- Removed) (+ Added) feature/aspect |
| ~09/19 | a0.01 | * + C# Console application * + using inline assembly to call needed instructions and retrieve data * + Information to be gathered from OS:   + AMD/Intel   + CPU model   + Native or Virtual System   + Method One Results   + Method Two Results   + Method Three Results |
| ~09/19 | a0.02 | * - + using FASM to build an assembly written DLL to call instructions (inline compilation may be unreliable). * - Inline assembly removed. * + DLL calls methods (pseudocode: measure, measure, code, measure) per procedure. |
| ~09/19 | a0.03 | * - + DLL calls sectioned into specific functions (no longer measures bias, code, and final measurement in a single continuous procedure). This lowers the measurement for bias, and code. Which should produce clearer results after gathering. * - single method procedures removed |
|  |  | * Thought has been given to a Debian/linux version of this application. |
| 06/10/19 | a0.10 | * + The state pattern has been applied to the console application, to create a better more understandable UX. * + States added: Initial Agreement/Explanation, Native/Virtual system choice, Results view/Send, thank you. * + Method One works reliably |
| 15/10/19 | a0.11 | * + Decision made to expand gathered information with direct from CPUID information. Expanded:   + CPUID 0H result (GenuineIntel/AuthenticAMD)   + CPUID 01H (family and model) |
| 16/10/19 | a0.11 | * ? Method Three implementation enacts too many instructions (Figure 1). And needs to be changed. |
| 18/10/19 | a0.12 | * - CPUID 01H (family and model) information discarded. (unable to interpret). Will use intel’s ARK, and AMD equivalent during analysis phase from OS reported model information. * -+ Method Three implementation changed. Now results in less calls, but slightly too high (above 3-4 in virtual machine) (Figures 2 and 3) |

## Deliverable: Testing and changelog

|  |  |
| --- | --- |
| Date | Changes Made (- Removed) (+ Added) feature/aspect |
| 22/02/2020 | * Intel/AMD inconsistencies Tabled with tests to perform |
| 26/02/2020 | * +- Changes made to DLL to assist in testing   + + Addition of CPUID 01H info and timing   + + CPUID 4 000 0000H info   + +- Name changes to reflect new functionality |
| 03/03/2020 | * + Implemented memory reading, writing, and read measuring procedures   + Procedures are retuning errors for writing to protected memory   + Read procedures are not returning the written uint32 values |
| 06/03/2020 | * +- DLL “uses” changed. Registers given do not need to be separated by a comma   + - Removing the comma allows reading and writing to occur as expected |
| 07/03/2020 | * Memory measurement seems to behave out of the ordinary.   + Instead of returning normal Minor TSC and Major TSC, returns Major TSC Minor TSC. Presenting as an “overflowing” the TSC.   + Overflow cannot always occur within the timeframe tested. Swapping Minor-Major these alleviates the issue   + Measurements have returned in the 25 million range for normal accession, and 33 million for eviction accession. Far outside of expected ranges. |

## Intel 64, IA-32 and AMD 64 Inconsistencies

Taken from the “Intel 64 and IA-32 Architectures Software Developer’s Manual” and the “AMD64 Architecture Programmer’s Manual”

|  |  |  |
| --- | --- | --- |
| **Instruction or Process** | **Intel** | **AMD** |
| CPUID 0H | EBX,ECX,EDX return of GenuineIntel | Same as intel but  AuthenticAmd returned |
| CPUID 01H  (only important information mentioned) | EAX:  Extended Family bits 27-20  Extended Model bits 19-16  Family bits 11-8  Model bits 7-4  Stepping bits 3-0  EBX:  Brand id bits 7-0  ECX:  VMX bit 5  Not used bit 31 always 0  EDX:  TSC bit 4  MSR bit 5 | EAX: No differences  EBX: No differences  ECX:  Bits 8-4 Reserved  RAZ bit 31 “Reserved for use by hypervisor to indicate guest status”  EDX: No differences |
| CPUID 4 000 0000H | 4 000 000 through 4 FFF FFFF are invalid | 4 000 0000 through 4 000 FF  “These function numbers are reserved for use by the virtual machine monitor” |
| IA\_32\_BIOS\_SIGN\_ID  Checking the revision of the loaded CPU microcode | Details within the manual  9.11.7.1 pg3050 Volume 3A Chapter 9  Revision returned in EDX | Requires ring 0 access. As such is not acceptable for implementation as it breaks “Not to require elevated or administrative permissions”. |

## Intel/AMD Inconsistency Testing

|  |  |  |
| --- | --- | --- |
| **Test** | **Methodology** | **Result** |
| CPUID 01H ECX return  VMX bit 5 | Implement method in DLL for bit checking. Test on both AMD and Intel system | Both return true. |
| CPUID 01H ECX  RAZ bit 31 “Hypervisor Present bit” | Implement method in DLL for bit checking. Test on both AMD and Intel system | Intel Development system returns true in both native and virtual. |
| CPUID 4 000 0000H  “VMWare hypervisor” leaf | Check on both native, VirtualBox, and VMWare virtualisation | EBX ECX EDX Results:  Native: 7 832 170700800  Virtual: 1073741830 2020557398 3089214592  Both are responsive, and usable for hypervisor brand identification |

<https://kb.vmware.com/s/article/1009458>

# Appendix E: AWS

## AWS Activity Diagram

A close up of a map

Description automatically generated

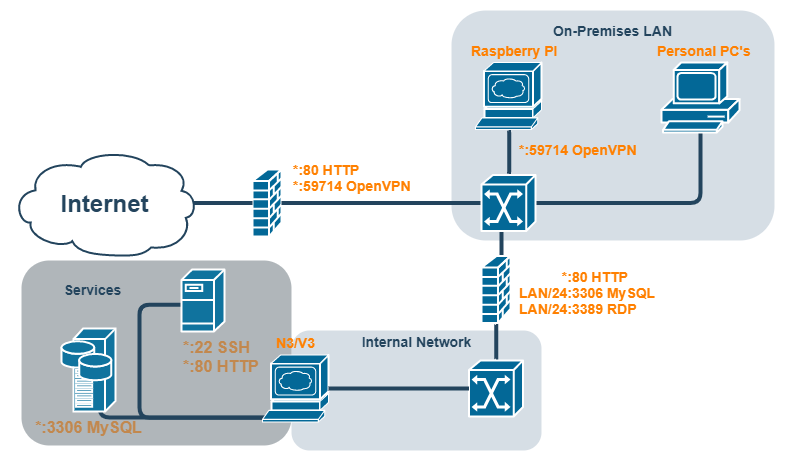
Utilising the LAMP (Linux, Apache, MySQL, PHP) stack

## AWS Network Diagram

A picture containing clock

Description automatically generated

## On-Premises Network Diagram



## JSON response codes and messages (AWS to Gatherer Application)

|  |  |
| --- | --- |
| **Code** | **Response** |
| 400 | “No data submitted” |
| 401 | “Data submitted is incorrect” |
| 402 | “Data failed verification” |
| 403 | “Data failed to store” |
| 200 | “Data Stored Correctly” |

# Appendix F: MySQL

## Prototype Normal Form Process

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0NF** |  |  |  |  | **3NF:** | **CPU\_Name** |
| CPUName |  | **CPU** |  | **CPUName** |  | Name |
| VMX |  | CPUName |  | Name |  |  |
| RAZ |  | Vendor |  |  |  | **CPU\_Prefix** |
| TSC |  | Family |  | **CPU** |  | Prefix |
| MSR |  | FamilyExt |  | Vendor |  |  |
| Vendor |  | Model |  | Family |  | **CPU\_Vendor** |
| Family |  | ModelExt |  | FamilyExt |  | Vendor |
| Model |  | Stepping |  | Model |  |  |
| FamilyExt |  | MCRevision |  | ModelExt |  | **CPU** |
| ModelExt |  |  |  | Stepping |  | **\*Name** |
| Stepping |  | **Result** |  | MCRevision |  | **\*Prefix** |
| MCRevision |  | \*CPU |  |  |  | **\*Vendor** |
| TimeAdded |  | TimeAdded |  | **Result** |  | Family |
| NOPMinima |  | VMX |  | \*CPU |  | FamilyExt |
| NOPMean |  | RAZ |  | TimeAdded |  | Model |
| NOPMode |  | TSC |  | VMX |  | ModelExt |
| NOPMaxima |  | MSR |  | RAZ |  | Stepping |
| CPUIDMinima |  | NOPMinima |  | TSC |  | MCRevision |
| CPUIDMean |  | NOPMean |  | MSR |  |  |
| CPUIDMode |  | NOPMode |  | NOPMinima |  | **Result** |
| CPUIDMaxima |  | NOPMaxima |  | NOPMean |  | \*CPU |
| EvictionMinima |  | CPUIDMinima |  | NOPMode |  | TimeAdded |
| EvictionMean |  | CPUIDMean |  | NOPMaxima |  | VMX |
| EvictionMode |  | CPUIDMode |  | CPUIDMinima |  | RAZ |
| EvictionMaxima |  | CPUIDMaxima |  | CPUIDMean |  | TSC |
| StealthMinima |  | EvictionMinima |  | CPUIDMode |  | MSR |
| StealthMean |  | EvictionMean |  | CPUIDMaxima |  | NOPMinima |
| StealthMode |  | EvictionMode |  | EvictionMinima |  | NOPMean |
| StealthMaxima |  | EvictionMaxima |  | EvictionMean |  | NOPMode |
|  |  | StealthMinima |  | EvictionMode |  | NOPMaxima |
|  |  | StealthMean |  | EvictionMaxima |  | CPUIDMinima |
|  |  | StealthMode |  | StealthMinima |  | CPUIDMean |
|  |  | StealthMaxima |  | StealthMean |  | CPUIDMode |
|  |  |  |  | StealthMode |  | CPUIDMaxima |
|  |  |  |  | StealthMaxima |  | EvictionMinima |
|  |  |  |  |  |  | EvictionMean |
|  |  |  |  |  |  | EvictionMode |
|  |  |  |  |  |  | EvictionMaxima |
|  |  |  |  |  |  | StealthMinima |
|  |  |  |  |  |  | StealthMean |
|  |  |  |  |  |  | StealthMode |
|  |  |  |  |  |  | StealthMaxima |

## Prototype EER Diagram

A screenshot of a cell phone

Description automatically generated

## Prototype Notes

* CPU Vendor,Name,Prefix can be taken from the Gathering Application, and programmatically entered.
* Microcode Revision placed in CPU to more easily distinguish between CPU revisions when analysing.
* VMX,RAZ,TSC,MSR placed in submission to more easily detect any manipulation, on a per submission basis.

## Final Normal Form Process

|  |  |  |  |
| --- | --- | --- | --- |
| **0NF** |  |  | **3NF** |
| CPUName | **CPU** | **CPUName** | **CPU\_Name** |
| Vendor | CPUName | Name | Name |
| Family | Vendor |  |  |
| Model | Family | **CPU** | **CPU\_Prefix** |
| FamilyExt | FamilyExt | Vendor | Prefix |
| ModelExt | Model | Family |  |
| Stepping | ModelExt | FamilyExt | **CPU\_Vendor** |
| OS | Stepping |  |  |
| NOPMinima |  | Model | Vendor |
| NOPMean | **Result** | ModelExt |  |
| NOPMode | \*CPU | Stepping | **CPU** |
| NOPMaxima | OS |  | **\*Name** |
| CPUIDMinima | NOPMinima | **OS** | **\*Prefix** |
| CPUIDMean | NOPMean | Name | **\*Vendor** |
| CPUIDMode | NOPMode |  | Family |
| CPUIDMaxima | NOPMaxima | **Result** | FamilyExt |
| TLB | CPUIDMinima | \*CPU | Model |
| TLBEviction | CPUIDMean | \*OS | ModelExt |
| StealthMinima | CPUIDMode | NOPMinima | Stepping |
| StealthMean | CPUIDMaxima | NOPMean | MCRevision |
| StealthMode | TLB | NOPMode |  |
| StealthMaxima | TLBEviction | NOPMaxima | **OS** |
|  | StealthMinima | CPUIDMinima | Name |
|  | StealthMean | CPUIDMean |  |
|  | StealthMode | CPUIDMode | **Result** |
|  | StealthMaxima | CPUIDMaxima | \*CPU |
|  |  | TLB | \*OS |
|  |  | TLBEviction | NOPMinima |
|  |  | StealthMinima | NOPMean |
|  |  | StealthMean | NOPMode |
|  |  | StealthMode | NOPMaxima |
|  |  | StealthMaxima | CPUIDMinima |
|  |  |  | CPUIDMean |
|  |  |  | CPUIDMode |
|  |  |  | CPUIDMaxima |
|  |  |  | TLB |
|  |  |  | TLBEviction |
|  |  |  | StealthMinima |
|  |  |  | StealthMean |
|  |  |  | StealthMode |
|  |  |  | StealthMaxima |

## Final EER Diagram

A screenshot of a cell phone

Description automatically generated

## Final Notes

* CPU Foreign keys not null deemed over engineering, and Impractical.
* CPU Vendor,Name,Prefix can be faked, and are not a reliable source of naming. CPUID vendor/family/model marginally more reliable. Will be selected, from Intel ARK and AMD equivalent by the database administrator.
  + CPU FK NN, mustn’t be done programmatically. As Errors likely with spoofed static environmental characteristics.
* VMX, RAZ, TSC, MSR removed as can also be faked. Further, RAZ unreliable, TSC needed to function, MSR not used, VMX detectable through analysis.
* OS added. Context May help analysis if the Operating system can be a factor. This is prefilled by the database administrator.
* CPU’s can be added before the submission is.

## MySQL Security

A screenshot of a cell phone

Description automatically generated

Diagram representing the write access to the database per account.

# Appendix G: Initial Testing

## Test System

The test system is a mid 2012 system consisting of the following core components:

An intel 3770k CPU, ASUS Maximus V Formula Motherboard

Other components are not listed as they are considered less relevant to the phenomena tested.

## Assembly

The pseudocode for measuring instruction speed using method one is as follows:

void MethodOne? (ref long[])

{

Clock\_Cycle Measurement;

Clock\_Cycle move to long[0];

Push long\_pointer;

Instruction;

Pop long\_pointer;

Clock\_Cycle Measurement;

Clock\_Cycle move to long[1];

}

## CPUID EAX testing

When executing the CPUID instruction with an entry in the eax register, different behaviours can occur. The cycle time for these was measured in order to find the most suitable candidate for use as a measurement device. The following measurements are the minima of 100 tests for each eax value specified in the Intel 64 and IA-32 Architectures manual, volume 2A, instruction set reference A-Z, Table 3-8, page 3-191. Anomalous entries’ purposes have been specified.

### Measured CPUID Instructions:

|  |  |  |
| --- | --- | --- |
| **Purpose of eax value** | **Eax value** | **Cycle Time (100 test minima)** |
| Basic CPUID Information: (GenuineIntel) | 0H | 230 |
| Basic CPUID Information: Version, feature set, etc | 01H | 349 |
| Basic CPUID Information: Cache | 02H | 280 |
| - | 03H | 230 |
| - | 04H | 230 |
| MONITOR/MWAIT Leaf | 05H | 283 |
| Thermal and Power Management Leaf | 06H | 754 |
| Structured Extended Feature Flags… | 07H | 254 |
| (not specified) | 08H | 248 |
| Direct Cache Access Information Leaf | 09H | 286 |
| Architectural Performance Monitoring Leaf | 0AH | 283 |
| Extended Topology Enumeration Leaf | 0BH | 289 |
| Processor Extended State Enumeration Main Leaf | 0DH | 316 |
| Intel RDT Monitoring…/L3 Cache Monitoring… | 0FH | 316 |
| Intel RDT…/L3 Cache…/L2 Cache… Allocation… | 10H | 316 |
| Intel SGX Capability Enumeration Leaf | 12H | 316 |

# Appendix H: Application for Ethical Approval

**PLYMOUTH UNIVERSITY FACULTY OF SCIENCE AND ENGINEERING**

**Research Ethics & Integrity Committee**

APPLICATION FOR ETHICAL APPROVAL OF RESEARCH INVOLVING

HUMAN PARTICIPANTS

**All applicants should read the guidelines which are available via the following link:**

<https://staff.plymouth.ac.uk//scienv/humanethics/intranet.htm>

This is a WORD document. Please complete in WORD and extend space where necessary. Clearly name any supporting documents and reference in the application.

*Postgraduate and Staff must submit a signed copy to* [SciEngHumanEthics@plymouth.ac.uk](mailto:SciEngHumanEthics@plymouth.ac.uk)

*Undergraduate students should contact their School Representative of the Science and Engineering Research Ethics & Integrity Committee or dissertation advisor prior to completing this form to confirm the process within their School.*

***School of Computing, Electronics and Mathematics******undergraduate students*** *– please submit to* [SciEngHumanEthics@plymouth.ac.uk](mailto:SciEngHumanEthics@plymouth.ac.uk) *with your project supervisor copied in.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ***TYPE OF PROJECT***

***1.1 What is the type of project?***

|  |  |  |
| --- | --- | --- |
| **Applicant** | **Type** | **Put X in 1 only** |
| STAFF | Specific project |  |
| Thematic programme of research |  |
| Practical / Laboratory Class |  |
| POSTGRADUATE STUDENTS | Taught Masters Project |  |
| M.Phil / PhD by research |  |
| UNDERGRADUATE STUDENTS | Student research project | **X** |
| Practical / Laboratory class where you are acting as the experimenter |  |

1. ***APPLICATION***

|  |
| --- |
| ***2.1 TITLE of Research project*** |
| Hardware Assisted Virtualisation Detection |
| ***2.2 Name, telephone number, e-mail address and position of applicant for this project (plus full details of Project Supervisor for postgraduate and undergraduate students)*** |
| Applicant: James Murray 07478771747 james.murray@students.plymouth.ac.uk Undergraduate Student  Supervisor: Marco Palomino (marco.palomino@plymouth.ac.uk) Lecturer in Information Systems and Big Data |
| ***2.3 General summary of the proposed research for which ethical clearance is sought, briefly outlining the aims and objectives (no more than 200 words)*** |
| This project aims to discover relationships between Computer CPU characteristics, to identify further areas of study and establish new information about the current state of CPU onboard virtualisation hardware.  Participants will allow a program to extract information from, and run tests on, their computer system. This program will ask for this to be sent to a cloud-based collection service, and will display the results of the tests. No personally identifiable information is collected (e.g. Serial Numbers, IP addresses, Names) only hardware characteristics (Processor name, revision, etc..) and test results.  Collected data will be analysed to find relations and new information regarding efficiencies and behaviour. |
| ***2.4***  ***Physical site(s) where research will be carried out*** |
| N/A Online only for participants |
| ***2.5 Does your research involve external institutions (e.g. other university, hospital, prison etc. see guidelines)*** |
| Delete as applicable: **No** |
| ***2.5a If yes, please give details:*** |
| ***2.5b If yes, you must provide letter(s) from institutional heads permitting you to carry out research on their clients, and where applicable, on their sites(s). Are they included?*** |
| Delete as applicable: **No Yes** |
| ***If not, why not?*** |
|  |
| ***2.6 Start and end date for research for which ethical clearance is sought (NB maximum period is 3 years)*** |
| **Start date: 17/02/2020 Date of approval End date: 17/04/2020** |
| ***2.7 Has this same project received ethical approval from another Ethics Committee?*** |
| Delete as applicable: **No** |
| ***2.7a If yes, do you want Chair’s action?*** |
| Delete as applicable: **No Yes** |
| ***If yes, please include other application and approval letter and STOP HERE. If no, please continue*** |

1. ***PROCEDURE***

|  |
| --- |
| ***3.1 Describe (a) the procedures that participants will engage in, and (b) the methods used for data collection and recording*** |
| 1. ***Participants will download the program to their system (virtual or physical) and run the program. The program will prompt for information regarding the system on which it is being run (Physical, Virtual host based, Virtual Hypervisor based). The program will then collect system information and test results. The program will then show the participant the information collected and tests results. The participant will be prompted for acceptance/denial of sending this information. The information will then be sent (if accepted) to my cloud-based website for storage, and then the program will close.*** 2. ***This information will be encrypted, sent to a cloud based website over http, which will decrypt and process this into a MySQL database for further analysis.*** |
| ***3.1a If surveying or interviewing, you must include your questionnaire(s) and interview schedule(s).***  ***Are these attached:***  Delete as applicable:**No** |
| ***This research does not survey or interview human participants, it only collects information from computer systems directly, which is sent with the permission of the human participant.*** |
| ***3.2 How long will the procedures take? Give details*** |
| ***Participants are expected to take no longer than 1 minute using this program. But may choose to utilise the program multiple times.*** |
| ***3.3 Does your research involve deception?*** |
| Delete as applicable: **No**  *If no go to section 4 If yes, complete section 3.3 a-d* |
| ***Please explain why the following conditions apply to your research:*** |
| ***3.3a Deception is completely unavoidable if the purpose of the research is to be met*** |
|  |
| ***3.3b The research objective has strong scientific merit*** |
|  |
| ***3.3c Any potential harm arising from the proposed deception can be effectively neutralised or reversed by the proposed debriefing procedures*** |
|  |
| ***3.3d Describe how you will debrief your participants*** |
|  |

***4. BREAKDOWN OF PARTICIPANTS***

***4.1 Summary of participants***

|  |  |
| --- | --- |
| ***Type of participant*** | ***Number of participants*** |
| *Non-vulnerable Adults* | *Estimated Minimum: 400* |
| *Minors (< 16 years)* | *None* |
| *Minors (16-18 years)* | *None* |
| *Vulnerable Participants*  *(other than by virtue of being a minor)* | *None* |
| ***TOTAL*** | *Estimated Minimum: 400* |

|  |
| --- |
| ***4.2 How were the sample sizes determined?*** |
| ***The number of samples needed to generate reliable insights is unknown. This is due to the relatively new area of research, and the exploratory nature of the project. The estimate in 4.1 is a conservatively estimated response rate based on the reported response rates of other studies, and the estimated reach of the advertisements.*** |
| ***4.3 How will subjects be recruited?*** |
| ***Participation in the program will be advertised on internet “boards” specific to virtualisation, hardware, and any related fields.***  ***This advertisement will direct participants to a website which hosts the program; gives instruction on it’s download and use within multiple environments; and further clarifies the research and gathered information.*** |
| ***4.4 Will subjects be financially rewarded? If yes, please give details.*** |
| ***No incentives will be given.*** |

***5. NON-VULNERABLE ADULTS***

|  |
| --- |
| ***5.1 Are some or all of the participants non-vulnerable adults?*** |
| Delete as applicable:  **Yes** |
| ***5.2 Inclusion / exclusion criteria*** |
| ***None.*** |
| ***5.3 How will participants give informed consent?*** |
| ***As the program is not installed, participants will give consent for information sharing directly within the program. After testing participants will be met with the results screen which includes a consent interaction. No information will be sent when this information is denied.*** |
| ***5.4 Consent form(s) attached*** |
| Delete as applicable: **No** |
| ***If no, why not?*** |
| ***Information is given through both the program and website from which the program is downloaded. See Annex A and attached current website progress for context.*** |
| ***5.5 Information sheet(s) attached*** |
| Delete as applicable: **No Yes** |
| ***If no, why not?*** |
| ***Information is given through both the program and website from which the program is downloaded. See Annex A and attached current website progress for context.*** |
| ***5.6 How will participants be made aware of their right to withdraw at any time?*** |
| ***Due to the anonymous nature of the project participants will not be able to withdraw submitted data. This is explained to participants before submission. see Annex A, Figure 1 and attached current website progress.*** |
| ***5.7 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?*** |
| ***The information gathered will be of an anonymous nature, and also wholly based upon the system of the individual. No personally identifying information is gathered. As such no participant will be identifiable from collected data.***  ***The data that is gathered will be stored in a MySQL database in the same AWS instance as the website. This MySQL database will be protected by a username and strong 16 character password. This MySQL database will initially only be internally accessible to the website. This will be periodically temporarily opened to external network access to download data for analysis. Upon project completion it will be downloaded one final time, and the Instance will be irrevocably deleted.*** |

***6. VULNERABLE PARTICIPANTS (Minors <18 years, and Vulnerable Adults)***

|  |
| --- |
| ***6.1 Are some or all of the participants:*** |
| (Delete as applicable)  ***Under the age of 16?*** **No**  ***Between the ages of 16 and 18?*** **No**  ***Vulnerable adults? (See guidelines)* No** |
| ***If no to all, please proceed to section 7.*** ***If yes, please continue and consult guidelines for working with minors and/or vulnerable groups.*** |

|  |
| --- |
| ***6.2 Describe the vulnerability (for minors give age ranges)*** |
|  |
| ***6.3 Inclusion / exclusion criteria*** |
|  |
| ***6.4 How will minors and vulnerable adults give informed consent?*** |
| ***Please delete as applicable and explain below (See guidelines)*** For minors < 16 only: **Opt-in Opt-out** |
| ***If opt-out, why?*** |
|  |
| ***6.5a Consent form(s) for minor/vulnerable adult attached*** |
| Delete as applicable: **No Yes** |
| ***If no, why not?*** |
|  |
| ***6.5b Information sheet(s) for minor/vulnerable adult attached*** |
| Delete as applicable: **No Yes** |
| ***If no, why not?*** |
|  |
| ***6.6a Consent form(s) for parent / legal guardian attached*** |
| Delete as applicable: **No Yes** |
| ***If no, why not?*** |
|  |
| ***6.6b Information sheet(s) for parent / legal guardian attached*** |
| Delete as applicable: **No Yes** |
| ***If no, why not?*** |
|  |
| ***6.7 How will parent/legal guardians, minors and/or vulnerable adults be made aware of their right to withdraw at any time?*** |
|  |
| ***6.8 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?*** |
|  |
| ***Investigators working with children and vulnerable adults legally require clearance from the Disclosure and Barring Service (DBS)*** |
| ***6.9 Do ALL experimenters in contact with children and vulnerable adults have current DBS clearance? Please include photocopies.*** |
| Delete as applicable: **No** |
| ***If no, explain*** |
|  |

***7. PHYSICAL RISK ASSESSMENT***

|  |
| --- |
| ***7.1 Will participants be at risk of physical harm (e.g. from electrodes, other equipment)? (See guidelines)*** |
| Delete as applicable: **No (Go to Q8)** |
| ***7.1a If yes, please describe*** |
|  |
| ***7.1b What measures have been taken to minimise risk?*** |
|  |
| ***7.1c How will you handle participants who appear to have been harmed?*** |
|  |

***8. PSYCHOLOGICAL RISK ASSESSMENT***

|  |
| --- |
| ***8.1 Will participants be at risk of psychological harm (e.g. viewing explicit or emotionally sensitive material, being stressed, recounting traumatic events)? (See guidelines)*** |
| Delete as applicable: **No (Go to Q9)** |
| ***8.1a If yes, please describe*** |
|  |
| ***8.1b What measures have been taken to minimise risk?*** |
|  |
| ***8.1c How will you handle participants who appear to have been harmed?*** |
|  |

***9. RESEARCH OVER THE INTERNET***

|  |
| --- |
| ***9.1 Will research be carried out over the internet?*** |
| Delete as applicable:  **Yes** |
| ***9.1a If yes, please explain protocol in detail, including how informed consent will be obtained, procedures concerning the right to withdraw and how confidentiality will be maintained. Give details of how you will guard against abuse by participants or others (see guidelines)*** |
| ***Participants will be given written information stating that their use of the program is voluntary, and that any information sent is not identifiable. A statement informing participants that the program will give an option to send data, will also be included.***  ***Participants will have the option to reject sending of system and test data during the programs use. Neither participants nor their systems will be identifiable from the sent information.***  ***Abuse of the submitted data is mitigated against using “White-Box cryptography”. It will be highly unlikely that a malicious actor will be able to submit incorrect information to the collection service.***  ***Please see attached visuals of the program and website.*** |
| ***9.1b Have you included the online version of questionnaire and information/consent form? This should be as close to the format which will be viewed on line as possible.***  Delete as applicable: **No** |

***10. CONFLICTS OF INTEREST & THIRD PARTY INTERESTS***

|  |
| --- |
| ***10.1 Do any of the experimenters have a conflict of interest? (See guidelines)*** |
| Delete as applicable: **No** |
| ***If yes, please describe*** |
|  |
| ***10.1a Are there any third parties involved? (See guidelines)*** |
| Delete as applicable: **No (Go to Q11)** |
| ***If yes, please describe*** |
|  |
| ***10.1b Do any of the third parties have a conflict of interest?*** |
| Delete as applicable: **No Yes** |
| ***If yes, please describe*** |
|  |

***11. ADDITIONAL INFORMATION***

|  |
| --- |
| ***11.1 Give details of any professional bodies whose ethical policies apply to this research*** |
|  |
| ***11.2 Please give any additional information that you wish to be considered in this application*** |
|  |

***12. ETHICAL PROTOCOL & DECLARATION***

To the best of our knowledge and belief, this research conforms to the ethical principles laid down by the University of Plymouth and by any professional body specified in section 10 above.

This research conforms to the University’s Ethical Principles for Research Involving Human Participants with regard to openness and honesty, protection from harm, right to withdraw, debriefing, confidentiality, and informed consent.

**Sign below where appropriate:**

**STAFF / RESEARCH POSTGRADUATES**

**Print Name Signature Date**

Principal Investigator: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Other researchers: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

**Staff and Research Postgraduates should email the completed and signed copy of this form to** [**scienghumanethics@plymouth.ac.uk**](mailto:scienghumanethics@plymouth.ac.uk)

**UNDERGRADUATE STUDENTS**

**Print Name Signature Date**

Student: JAMES MURRAY \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor / Advisor: MARCO PALOMINO \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

**Undergraduate students should pass on the completed and signed copy of this form to their School Representative of the Science and Engineering Research Ethics Committee.**

**Signature Date**

School Representative on Science and

Engineering Faculty Research Ethics & Integrity

Committee \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

**Faculty of Science and Engineering Research Ethics & Integrity Committee List of School Representatives**

School of Geography, Earth and Environmental Sciences Dr Sanzidur Rahman

School of Biological and Marine Sciences Dr Gillian Glegg (Chair)

Dr Victor Kuri

School of Biomedical Sciences Dr David J Price

School of Engineering Dr Asiya Khan

Mr Chris Pollard

School of Computing, Electronics & Mathematics Dr Mark Dixon

Dr Yinghui Wei

Doctoral College, Deputy Director Prof Steven Furnell

External Representative Dr Satish B K

Lay Member Rev. David V. Evans

### Committee Secretary: Mr Steven Neal

### Email: steven.neal@plymouth.ac.uk

### Tel: 01752 584877

## Annex A: Visuals and Evidence

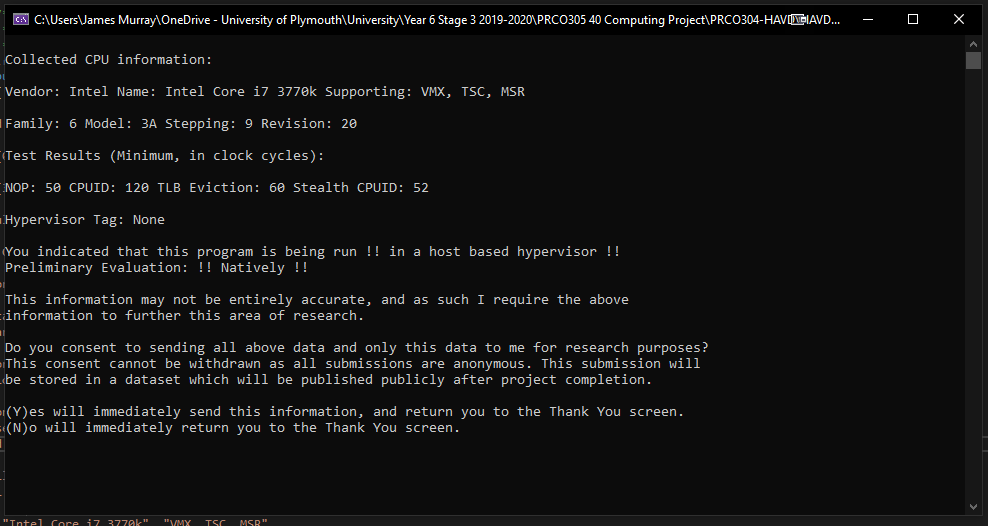


Figure 1 Results and Consent Screen

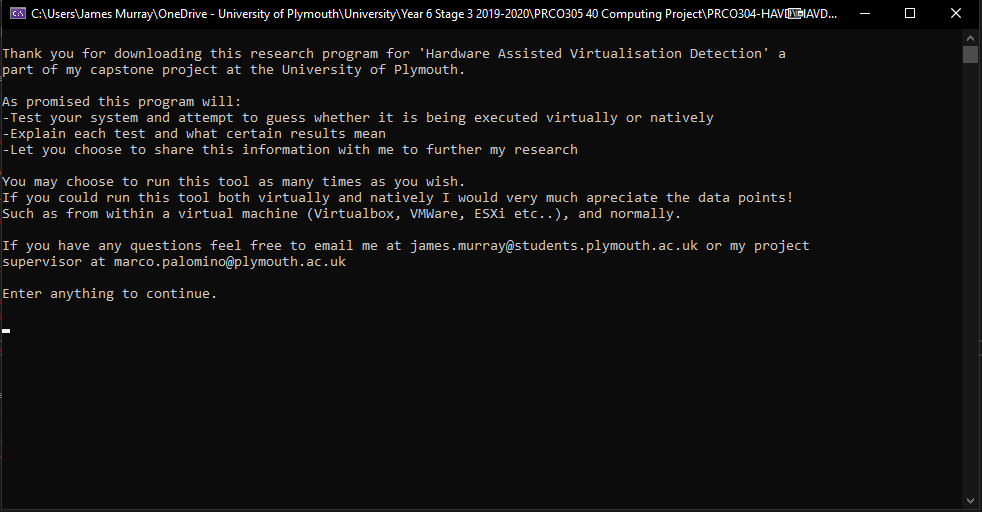


Figure 2 Initial Screen

A screenshot of a computer

Description automatically generated

Figure 3 Second Screen (hypervisor choice)

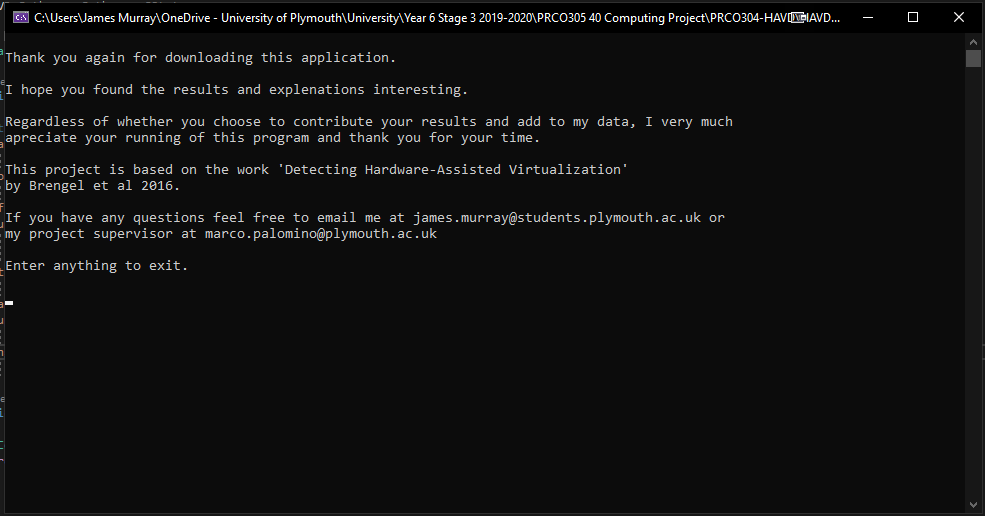


Figure 4 Thank you (final) Screen