University of South Wales
Prifysgol De Cymru

Faculty of Computing, Engineering and Science
M.Sc. Project

Avoiding, Preventing, and Mitigating CPU-based Side Channel Attacks: A Best Practice Guide

Thomas Phillip Harris, BSc. AMBCS.

18044611

First Supervisor: Dr. Richard Ward
Second Supervisor: Peter Eden
Year of Study: 2022
Scheme: M.Sc. Computer Systems Security

University of South Wales
Prifysgol De Cymru

Faculty of Computing, Engineering and Science

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described in this project is the result of the investigation carried out by the student, and that neither this project nor any part of it has been presented, or is currently being submitted in candidature for any award other than in part for the M.Sc. award, Faculty of Computing, Engineering and Science from the University of South Wales.

Signed: Thomas Phillip Harris

Abstract

The aim of this project is to perform extensive research on a variety of side-channel attacks, their effects, and how they are mitigated. Using this large pool of information, the attacks and their mitigation techniques will then be further explained using existing (secondary) and email/conversational based interviews with others (primary). Furthermore, there will be primary resources created for this project, in the form of a guide and interactive tutorial which allow academics (students, lecturers, and researchers) to broaden their knowledge on side-channel attacks and their mitigation techniques. The guide will be in a written format which is available online within both a GitHub repository and on a website that is also hosted on GitHub. Alongside this guide will be an interactive tutorial created using one of many available tools to provide a range of effective learning for the target audience.

The purpose of this project is to gather existing research on various CPU-based side-channel attacks and their mitigation strategies into a single source (referred to as a Best Practice Guide, or BPG). This source will be created as the deliverable for this project and will be suited towards academia to be used as a teaching aid. The BPG will contain a variety of both secondary (peer reviewed articles and journals) and primary sources (contact with researchers within the field). The BPG will be distributed in several ways, physically as a paper copy to allow hands on and portable use, online hosted on a previously constructed website to allow access at all times (internet access willing), and in an interactive format using a popular learning service called ‘TryHackMe’. This allows the creation of *rooms* that can be used to teach a particular cyber security technique or software. Using TryHackMe, a room will be created with brief information on side-channels attacks in general, with more readily available sub-tutorials on a specific attack (Spectre or one of its variants). Using these, users would learn the workings of a side-channel attack from the safety of a sandboxed and isolated VM that accompanies the room. There will be various objectives and ‘quests’ for users to fulfil that move forward with the end goal of embedding an understanding of side-channel attacks in a practical sense.

The predicted conclusion of this project will be a fully finished and comprehensive report and BPG on five CPUd side-channel attacks and their

mitigation techniques. The BPG being an effective teaching aid and referenceable document for future research. Furthermore, the BPG will have several forms to allow different individuals and academia (lecturers, students, researchers) to have access should they be offline, online, or away from their PC/laptop.

Acknowledgements

I would like to thank the University of South Wales for their support in everything I did regarding this course and my further career. I would also like to thank my lecturers for the continued support and communication they have provided throughout this year, they have been invaluable despite the continuing upheaval due to Covid-19. Furthermore, I would extend a special thank you to Dr. Richard Ward, who has gone above and beyond to ensure work I have been carrying out has been to an extremely high standard, he has also pushed me towards a career opportunity with the University and as such I am now employed by USW, which is greatly aiding my dissertation and my growth as a computer and cyber security professional.

I would also like to thank those within my personal life, particularly my family who have always pushed me to be the best I can, as well as providing whatever support I may have needed throughout my time in the course. My partner Shelby should have special mention for being a steady ground for me to return to when I began to struggle, her continuing support has allowed me to step back when I have been overwhelmed, ensuring I am able to return to the work and excel where I had once struggled.

Table of Contents

1.0 Introduction 1

1.1 Aims and Objectives 2

1.2 Table of Figures 3

2.0 Literature Review 4

3.0 Methodology 16

4.0 Side-Channel Attacks 20

4.1 Power-Analysis Attack 20

4.2 Cache-Based Attack 24

4.3 Time-Based Attack 26

4.4 Differential Fault Analysis Attack 28

4.5 Acoustic Cryptanalysis Attack 29

5.0 Best Practice Guide 31

5.1 Planning and Design 31

5.2 The Guide 31

5.3 Deliverable Evaluation 32

6.0 Conclusion 34

References 37

Appendices 43

1.0 Introduction

Each chapter within this report will have a topic, each with their own sub-topics. This first chapter (Introduction) contains the aims and objectives of the project, the outline of what research is primary (carried out by the project lead) or secondary (gathered research from peer-reviewed sources), and the description of content within each chapter.

As this project leans towards the study of existing research carried out by other researchers in the field, the majority of the work within the report and the BPG will be sourced from other studies, such as peer-reviewed journals and articles. There will be some primary research carried out in the form of contact with other researchers studying the same subject area. The BPG will be created using a culmination of both primary and secondary research with references provided where appropriate. This BPG will be created by the project lead and have language and structure suitable for academic use.

1.1 Aims and Objectives

1. Research and report upon a variety of side-channel attacks, their effects, and how they are mitigated.
    a. Study and understand a significant number of peer reviewed primary and secondary sources.
    b. Of these sources, understand the workings of CPU-based side-channel attacks along with their mitigation strategies.
    c. Author a report on the topic of side-channel attacks and their mitigation strategies.
2. Select five appropriate CPU-based side-channel attacks using suitable terminology.
    a. Evaluate chosen CPU-based side-channel attacks and justify their mention based on scope.
    b. Ensure all information pertaining to technical areas are suitable for those with no understanding of CPU-based side-channel attacks can follow the research that has been done.
    c. Evaluate methods to distribute the report through a most suitable channel, such as online (hosted on a website), and in paper format.
3. Create a best practice guide (BPG) which will inform academia on the various side-channel attacks, and their mitigation techniques.
    a. Provide a brief overview of what a side-channel attack is, along with the more specific CPU-based side-channel attack.
    b. Use five main CPU-based side-channel attacks as the base for the BPG.
    c. Describe in detail each attack, their effects, and their mitigation techniques.
    d. Create and publish a TryHackMe ‘room’ to allow users to try a side-channel attack (a variant of Spectre) on a Linux VM without fear of damaging their systems.

1.2 Table of Figures

Figure 1 Transient.fail (2022) diagram of Spectre and Meltdown 14
Figure 2 Transient.fail (2022) brief overview of Spectre-type attacks along with relevant references 15
Figure 3 Adapted Waterfall model used in the context of the planning and management of the project 21
Figure 4 Le et al (2008) equation to calculate the secret key through a Mono-bit Differential Power Analysis 25
Figure 5 Graphical representation of both correct and incorrect key guesses 26
Figure 6 Example used in Le et al (2008) (referred to as Figure 11) with a scenario using a signal size of L = 100. This represents the distribution of templates both in a real sense and through the use of the MD and ML metrics. 27
Figure 7 Framework diagram created for this report using the model created and utilised by Bulck et al (2017) (referred to as Figure 1) which outlines the process of Single-Stepping SGX enclaves 29
Figure 8 Genkin et al (2017) Fig.5 Acoustic measurements of various CPU operations recorded using a spectrogram 34

2.0 Literature Review

The security of a device, be this a desktop computer or mobile phone, is integral in protecting the privacy of those who use them. While many security issues are related to software accessing sensitive information related to a person or a system, hardware related issues can be just as damaging, if not more so.

A CPU-based side-channel attack (which includes cryptographic systems), as defined by Su and Zeng (2021 p.1), is an attack which “utilize shared CPU caches within the same physical device to compromise the system’s privacy (encryption keys, program status, etc.)”. This is an attack that uses a misconfiguration of the firmware or microarchitecture of a CPU to leak information that the CPU has access to through a *side-channel* or *cache* (although these attacks are also commonly used on cryptographic devices and systems that also use cryptographic keys). There have been several journals and peer-reviewed papers detailing studies on this topic which will be described and explained in detail throughout this paper, as well as creating a Best Practice Guide for academia to use in their teaching of this more specific type of attack.

While there are many distinct types of side-channel attacks that target various different aspects of a computer system, this paper looks into the more specific CPU-based side-channel attacks, particularly how they occur, the scale of damage they can cause, and possible mitigation strategies (if any exist). An understanding of the basic concept of a CPU-based side-channel attack is essential before going in-depth on the various attacks. The cache of a CPU is a storage device, structured in such a way that enables it to be extremely fast at reading and writing (when compared to memory), although this means it is smaller than memory and as such the verbosity of the data stored within is lower. When a CPU uses any form of data, it first calls the cache to ensure this data isn’t available already, if it is this is considered a *cache hit* and if it is not available in the cache, this is a *cache miss*. Knowing this clarifies the concept of a side-channel attack as one that uses CPU firmware misconfiguration to access information that is present in the cache, allowing an attacker to steal its contents.

The first concept of a CPU-based side-channel attack was first considered by Percival (2005) who stated that the shared memory cache (the CPU cache) ‘permits a malicious thread (operating, in theory, with limited privileges) to monitor the execution of another thread, allowing in many cases for theft of cryptographic keys’. In his paper, Percival (2005) first introduces the ability to steal information (particularly the cryptographic keys of a system) from the CPU cache by running a malicious process on the targeted system. This process reads and records the ‘footprint’ created by OpenSSL which is recorded in the cache and as each set of calculations of both multiplications and squaring’s occur, each leaves a different ‘footprint’ of bits. Percival goes on to state that ‘in the case of OpenSSL we can typically identify the multiplier to within two possibilities in 50% of the modular multiplications.’, which, in simpler terms, means the secret cryptographic key used within OpenSSL can be calculated just by collecting the ‘footprint’ of bits stored in the CPU cache. Within his paper, Percival (2005) also put forward several solutions or mitigations to this danger, although this section will only include the most effective (to avoid the unsuitable or unrealistic solutions). One solution suggested was the application of the kernel scheduler, as stated by Percival (2005):

> ‘Recognizing that a side-channel between threads is only dangerous if the threads are operating at different privileges — or, put another way, if the threads are not permitted to debug each other — the scheduler could be written in such a way as to use the credentials of threads in the process of determining which threads should be scheduled with which (virtual) processors’

By doing this the threads that are not authorised (incorrect credentials) will not be scheduled on each processor and therefore not have the access to the data within the CPU cache (as they would neither have the need nor credentials to access it). Percival (2005) goes on to say that this method would have its drawbacks, particularly regarding performance as the kernel needs to verify and evaluate the threads to ensure they have the permission and credentials to share a processor core at several points (to further ensure that the side-channel is allowed to be created).

There are numerous attacks that take advantage of the CPU firmware which is what this paper will be focusing on. Of the examples of side-channel attacks, there are five which will be the focus on both the method of performing this attack, as well as the different mitigation strategies (if any) for each. The previous two attacks mentioned above are referred to as *cache-based side-channel attacks* and *timing attacks,* respectively. Further attacks that will be mentioned are the *power-analysis attacks*, *allocation-based attacks*, *acoustic cryptanalysis attacks*, and *Thermal Attacks*. Alongside these, infamous attacks that have recently made the news will have special mention, these being Spectre and Meltdown (for cache-based attacks) and the recent Hertzbleed (which is part of a newly conceptualised family of attacks, *frequency side channels*, as of 2022).

In general, a Power Analysis attack involves an attacker analysing the power consumed by a cryptographic device, and in knowing the various changes in voltage within the device (which occur through small movements of electrical currents), attackers can then learn small pieces of information about the data being manipulated by the cryptographic device. There are two types of Power Analysis attacks, *Simple Power Analysis* (*SPA*), and *Differential Power Analysis* (*DPA*), both involve some form of monitoring the power consumption to learn information, but the first (SPA) uses a visual examination of the current used by a device over a period of time, whilst the second (DPA) monitors this same power consumption but rather than monitoring just the device as a single entity, it monitors various cryptographic operations within a single cryptographic device (Kocher *et al*, 1999, p. 2-5). There is also a further sub-attack of the Power Analysis variety called a *Correlation Power Analysis* (*CPA*), this involves monitoring the power consumption of a cryptographic device (focusing on one particular cryptographic method), having the victim encrypt multiple plaintext data sets and recording the power consumption of each, begin attack the secret key in small parts (referred to as subkeys), and then using the closest guessed subkeys, piece together a full correct secret key. Luo *et al* (2018) carried out a study that used a CPA attack to recover data that was stored on a GPU. A GPU was targeted as they state, ‘they have been redesigned and are used to accelerate a wide range of applications, including deep neural networks, image reconstruction and cryptographic algorithms.’ In their study they were able to recover the ‘last round of key bytes of an AES (Advanced Encryption Standard) implementation on an NVIDIA TESLA GPU’ and because of this study it showed

clear need for the strengthening and hardening of GPUs against threats such as side-channel attacks.

Side-channel attacks can also be carried out by monitoring the number of resources assigned to a process (rather than the resources that have been used); these are referred to as *Allocation-Based Side-Channel attacks*. Angel *et al* (2020) have carried out a study that discovered a vulnerability that allows attacking the part of the CPU responsible for allocation resources to a process (in the case of their study, the MPM or *Metadata-Private Messenger*). By attacking this element of the CPU, Angel *et al* (2020) proved existing MPMs 'vulnerable to traffic analysis'. This not only shows the danger in the leaking of any data via a side-channel, but also shows that many existing systems are vulnerable to a side-channel attack, allowing information that may be classified or extremely sensitive in nature to be read by anyone with the capability of performing this type of side-channel attack. Angel *et al* (2020) also provide a mitigation technique through the form of PRAs (Private Resource Allocators) which is defined in the paper as 'a new cryptographic primitive' that allows the allocation of resources to all clients 'without revealing to the clients whether any other clients received resources'.

In a similar concept as the *Power Analysis Attack*, the *Thermal Attack* uses the temperature of a CPU against itself, particularly when this temperature is monitored by a thermal interface that unprivileged users have access to. Kim and Shin (2022) implemented their own version of a *Thermal Attack* called 'ThermalBleed', this attack targeted the KASLR (Kernel Address Space Layout Randomisation) within Linux systems. The KASLR simply randomises the location of the Kernel code within the memory each time the system boots (Daniel Lopez Azana 2017). Kim and Shin (2022) had predicted that it would be possible to 'distinguish between a cache hit and a physical memory access in memory load operations', this means there is a proven method (using thermal analysis) of telling address translations that hit a TLB (Translation Lookaside Buffer) from those that miss. Remzi and Andrea (2018) state that the TLB stores 'recent translations of virtual memory to physical memory. It is used to reduce the time taken to access a user memory location.,' and by distinguishing the difference between a hit and a miss on the TLB, Kim and Shin (2022) state that they are able to 'infer the TLB status on *core 0* from *core 5* in an i7-8700'. By having this inference, by understanding the TLB status on one core within the CPU, the others can be inferred without the need for

one-by-one analysis. This study shows clear evidence for the need for necessary security protections to be in place when considering CPU core temperature status and other software used to monitor CPU temperature, without said security protections the use of any of this software pose a security risk particularly with *Intel* CPUs. The study goes on to state that the best method of mitigating this vulnerability is preventing any non-privileged used from accessing the CPU temperature by requiring admin access to access the thermal interface.

Acoustic Cryptanalysis is a form of attack that uses the sound a CPU makes as it performs tasks and procedures as a method of understanding the type of cryptographic process performed, along with different patterns of operations and memory access. Shamir and Tromer (2007) state in their thesis, 'RSA signature/decryption sounds different for different secret keys', the two prove this by carrying out an attack on modern CPUs (modern at the time) and were able to 'exploit acoustic emanations from modern computers, wherein the power circuitry creates vibrations that are modulated by CPU activity'. They had performed this attack against the RSA cryptographic computation, which allowed leakage of sensitive information on the RSA computation which, in their own words, 'provide strong evidence that key recovery may be possible.' Within their paper, Shamir and Tromer (2007) also detail mitigation techniques, 'mitigation can be achieved through algorithmic techniques' and the implementation of 'careful circuit design and use of high-quality components, or by through shielding and physical access controls'. Of these techniques, the only method relevant to a CPU consumer (rather than a manufacturer) would be the shielding/physical access controls, a consumer is unable to determine the materials/components used in the creation of a CPU, or in fact the design of the circuitry on which the CPU works.

The concept of side-channel attacks is one that little is known about. Even now, as of writing this paper, new attacks have been researched and devised. One research group (Wang *et al* 2022) has devised a new family of side-channel attacks which they describe as evolving from *Power side-channel attacks* into their newly coined *Remote Timing attacks*. In their paper they state that 'in modern Intel (and AMD) x86 CPUs, DVFS-induced frequency variations depend on the current power consumption, and hence the data being processed.'. The attacks devised by the group, when run against a post-quantum cryptographic method called SIKE (Supersingular Isogeny Key Encapsulation) developed by Jao *et al* (2017), could

perform a ‘full key recovery attack through *remote timing*’. Alongside the attack, the authors of the paper also outlined a method of mitigating this form of attack by ensuring the thermal and/or power limits are not hit (as the reaching of these limits is when the DVFS system kicks in). To stop these limits being hit, Wang *et al* (2022) suggested turning off Turbo-Boost and other similar functionalities from the BIOS screen. While the previous mitigation strategy works in relation to CPUs, Wang *et al* (2022) had also outlined one that refers specifically to ciphers in which they suggest ‘removing secret-dependent leakage in cryptographic software’, which in relation to SIKE, prevents attacker-based ‘ciphertexts from triggering secret-dependent computations on 0s’. In their paper, Wang *et al* (2022) state that this exploit is present on a range of CPUs (both Intel and AMD) from 11th Gens, down to older 6th and 7th Gens.

There are several specific attacks from the *Cache-based side-channel attack* family, specifically Spectre and Meltdown, which target the data being passed through the cached memory of other programs. Both attacks differ slightly by both target the memory cache of a system (this being the side-channel) and attempt to recover information stored in the cache at the time of the attack. Both attacks can retrieve any information, be it passwords used by a password manager, or stored credit cards in a software wallet, from the CPU cache without the user's knowledge. Figure 1 is a branching diagram of both Meltdown and Spectre created by Graz University (2020), showing each attack, their variants, and the differing strategies each variant can use. Figure 2 is a further example of the information available on the diagram by Graz University (2020) which shows an explanation of whichever tab has been clicked on, along with appropriate references and diagrams (if applicable).

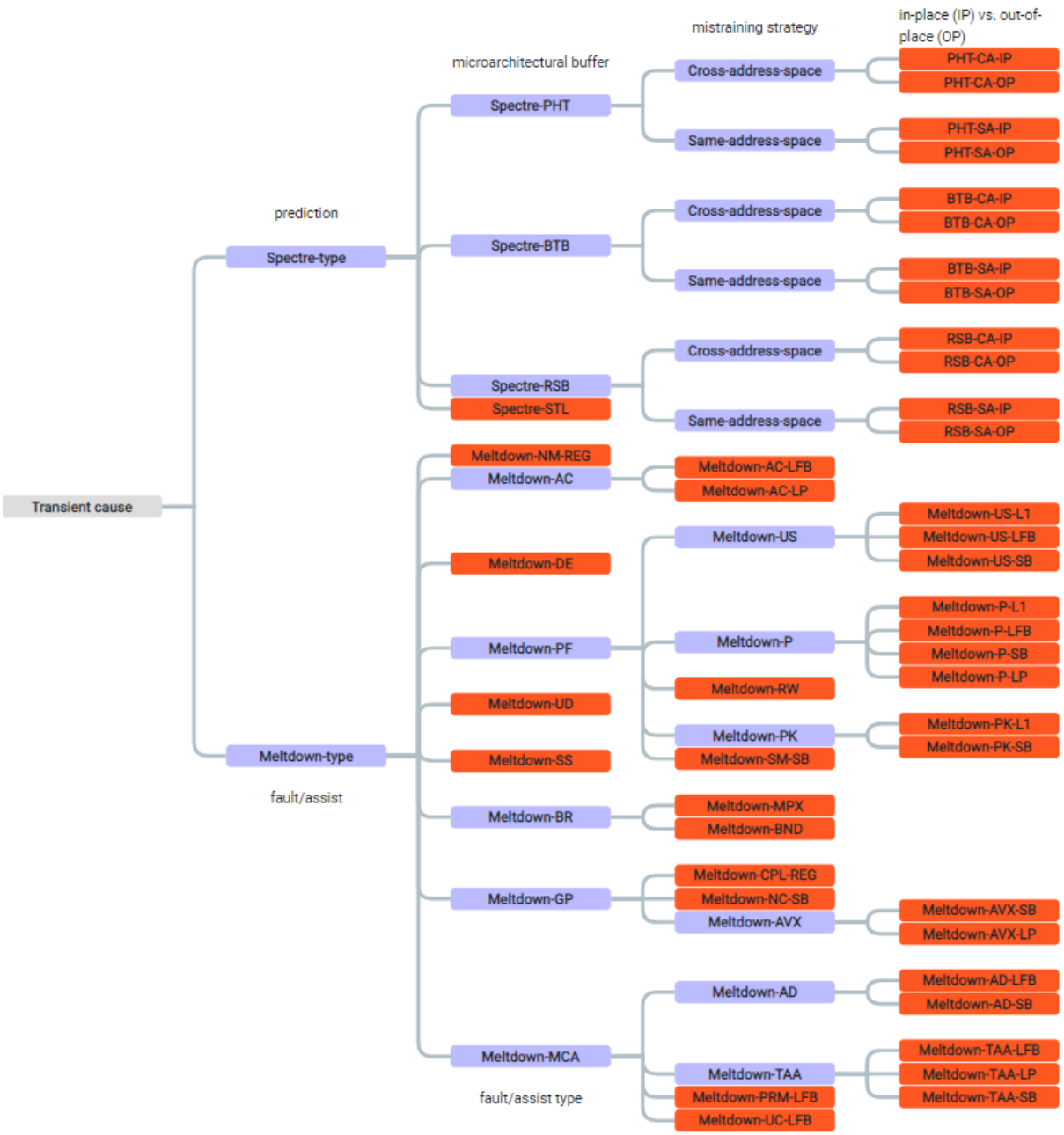


Figure 1 Transient.fail (2022) diagram of Spectre and Meltdown

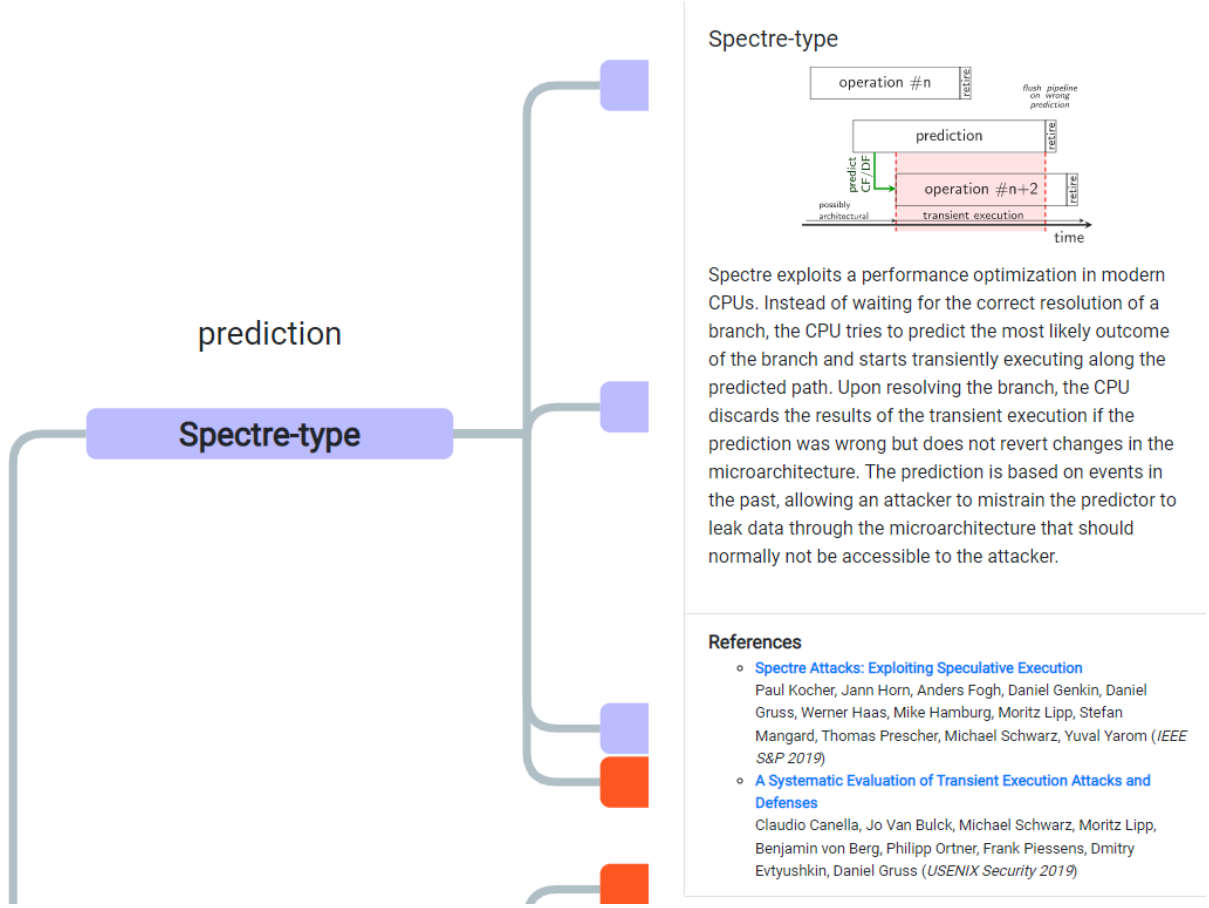


Figure 2 Transient.fail (2022) brief overview of Spectre-type attacks along with relevant references

Horn *et al* (2018) carried out a study on Meltdown, which they described as an attack that 'exploits side effects of out-of-order execution on modern processors to read arbitrary kernel-memory locations including personal data and passwords.'. This attack targets the performance feature called 'out-of-order' execution which 'schedules subsequent operations to idle execution units of the core' which improves the speed at which a CPU can run processes. As shown by the study (Horn *et al* 2018), the out-of-order execution feature has negative connotations through the potential of leaking information due to timing differences with the execution of both sequential and out-of-order execution. Horn *et al* (2018) further states that an attacker can 'dump the entire kernel memory by reading privileged memory in an out-of–order execution stream'. Horn *et al* (2018) also discusses some countermeasures, both in hardware and software form, which can mitigate or prevent Meltdown. One hardware mitigation they begin with is 'to disable out-of-order execution' although they go on to state that this would have a significant negative impact on the performance of the CPU. A further, more realistic, hardware mitigation

strategy they suggest is ‘to introduce a hard split of user space and kernel space.’ This mitigation would ensure that the user space resides in the lower half of the address space whilst the kernel space resides in the upper half, allowing any memory fetch to determine what request would constitute a violation of the security boundary ‘without requiring further lookups,’ these being checks the CPU would make against the requests using existing security boundaries.

Spectre differs from Meltdown as it targets the branch predictor and speculative execution aspects of a CPU and exploits these to execute operations that do not usually occur during normal program execution. These operations are executed as they are incorrectly predicted by the branch predictor which then leads to them being speculatively executed, which can then allow leaking of information via the cache (again, referred to as a side-channel). Kocher *et al* (2018) state in their paper that ‘At a high level, Spectre attacks trick the processor into speculatively executing instruction sequences that should not have been executed under correct program execution.’ Kocher *et al* (2018) have shown that is possible to use what they call *transient instructions*, which they define as instructions whose effects on the ‘nominal CPU state are eventually reverted’, and the influence on what transient instructions are speculatively executed, it is entirely possible to leak whatever information is stored within the victim’s memory address space at the time of execution. Spectre is delivered via a small program which can be run as a standalone program or as a parallel program, in the paper, Kocher *et al* (2018) created a small program (as a PoC) which delivers a secret message and stores it in the cache memory, which is then accessed and retrieved by a malicious attacker program, allowing them to view the secret message purely by reading the sequence of instructions within a process. The sequence of instructions, as Kocher *et al* (2018) explains, ‘acts as a covert channel transmitter that leaks the victim’s memory or register contents.’.

To remain within the scope of this paper, only the general description of Spectre attacks is used, however Kocher *et al* (2018) does go into further detail on the different variants of Spectre and how they can be used against a CPU. Finally, a number of mitigation strategies are discussed at length along with their advantages and disadvantages regarding performance of a system that may implement them. The first is disabling speculative execution which stops Spectre at the root, although Kocher *et al* (2018) does state that this would cause ‘a significant degradation in the

performance of the processor.'. A further mitigation strategy outlined which is more effective at preventing JavaScript based Spectre attacks is to separate the processes from the secret data, like the way Google Chrome isolates each site into separate processes (Chromium 2018), which would prevent a process that is affected by Spectre from accessing secret data in use by another. One theoretical mitigation strategy as mentioned by Kocher *et al* (2018) is the prevention of secret data (which is accessed via speculative execution) being accessed or used by operations that may leak it. One mitigation strategy discussed that is not considered particularly effective is under section VII subsection D of the paper by Kocher *et al* (2018), Limiting Data Extraction from Covert Channels which mitigation has been implemented by various major web browsers which have 'degraded the resolution of the JavaScript timer, potentially adding Jitter' (Microsoft Blogs 2018) to mitigate the attack. Jitter is a concept within telecommunications and electronics which allows 'deviation of true periodicity of a presumably periodic signal' (Wolaver D H 1991), this will prevent the acquisition of timing sources (which provide the information which leads to secret keys and other such information being leaked). This mitigation strategy is not as effective as previous ones however as, 'while this approach may decrease attack performance, it does not guarantee that attacks are not possible.' Preventing Branch Poisoning is the final mitigation strategy discussed by Kocher *et al* (2018) which was implemented by both Intel (Intel 2018) and AMD (AMD 2018) through the extension of a mechanism that had three alternate controls for controlling the indirect branches and their interaction with the data and processes in use. The first, called Indirect Branch Restricted Speculation (IBRS), 'prevents indirect branches in privileged code from being affected by branches in privileged code from being affected by branches in less privileged code.' The second mitigation, called Single Thread Indirect Branch Prediction (STIBP), 'restricts branch prediction sharing between software executing on the hyperthreads of the same core.' Finally, the third mitigation strategy, called the Indirect Branch Predictor Barrier (IBPB), 'restricts branch prediction sharing between software running before setting the barrier from affecting branch prediction by software running after the barrier.'

BranchScope (identified as CVE-2018-9056), new type of side-channel attack that targets the directional branch predictor (sharing similarities with Spectre), has been presented by Evtyushkin *et al* (2018) as an attack that relies on an attacker inferring 'the direction of an arbitrary conditional branch instruction in a victim

program by manipulating the shared directional branch predictor.' As standard in modern CPUs and microprocessors, the Branch Prediction Units (or BPUs) are used to ensure instruction delivery is not interrupted across a number of conditional branches (branches whose instructions are executing, pending a 'condition' that needs to be met). As more operations and processes can be executed on a single physical core, a BPU is required to sustain the previously mentioned instruction delivery across multiple condition branches. In general, side-channel attacks targeting the branch predictor focus on one aspect of the BPU, the Branch Target Buffer (BTB) which relies on the conditional branch being satisfied, which then has its target updated (the predicted instruction to be executed and the result of the previous), allowing an attacker to see 'whether or not a particular victim branch is taken' (Evtyushkin *et al* 2018). In BranchScope, the directional predictor is targeted as the location from which to leak information, which allows an attacker to perform a side-channel attack that targets branch predictors even with protections being placed on the BTB. BranchScope can overcome a natural mitigation that modern CPUs provide (this being the 'unpredictability of the complex hybrid prediction mechanisms') by generating only one-level predictions regardless of the multi-level predictors being in place within the processor. Furthermore, Evtyushkin *et al* (2018) go on to show that BranchScope is not limited by other mitigations that are in place to defend the BTB from side-channel attacks.

Evtyushkin *et al* (2018) developed a number of mitigations that are implemented both in software and the hardware to harden the relevant systems against BranchScope by providing BPUs that are secured against side-channel attacks. The first of these mitigations is a technique called 'if-conversion' (Choi *et al* 2001) which an optimization method for compilers that will change any conditional branch used into 'sequential code using conditional instructions such as cmov' which removes conditional branch instructions, entirely invalidating the BranchScope attack. As a minor note, this mitigation technique also displayed a capability to mitigate timing-based side-channel attacks as well, particularly with its ease of implementation. Alongside this software-based mitigation, some hardware mitigations have been proposed, one such mitigation being 'partitioning the BPU' to ensure that the victim and attacker structures are not shared, this is more relevant when an SGX enclave is targeted as those and normal code will then have distinct and separate branch predictors, preventing malicious code from affecting the branch

predictor, which would lead to leakage of information from the SGX enclave. Finally, the ‘Randomization of the PHT’ (PHT referring to the Pattern History Table) prevents collisions (the predictability of these being the primary requirement for BranchScope) by modifying the indexing functionality of the PHT to ensure some data received by the software entity is unique (the software entity, for example, could be ‘part of the SGX hardware state, or simply some random number generated by the process’). By ensuring the randomization is periodic, both BranchScope and probing attacks are prevented (although as Evtyushkin *et al* (2018) state, this does cause some performance degradation).

RetBleed is a newly discovered variant of the speculative execution side-channel attack ‘Spectre,’ this attack is relevant on both x86-64 and ARM processors (with particular focus on more modern and recent Intel and AMD chips). Registered under CVE-2022-29901, this vulnerability was discovered and researched by Wikner and Razavi (2022). As stated in their paper, Retbleed is a ‘Spectre-BTI attack that leaks arbitrary kernel memory on fully patched Intel and AMD systems’ which they say is made possible by two discoveries, the first being that ‘return instructions behave like indirect branches under certain microarchitecture-dependant conditions’ and the second being the ability of an attacker can ‘arbitrarily control the predicted target of such return instructions’. The name Retbleed comes from the already implemented mitigation for other Spectre variants (notably variant 2, Branch Target Injection, or BTI, attacks) called Retpoline which is marketed as a protection that ‘converts vulnerable branches into protected returns’ (Quinn 2021), otherwise known as the mitigation to stop Spectre. With this attack being able to bypass Retpoline, Wikner and Razavi (2022) proposed new mitigations, one of which was to prevent speculation, which has been further developed by AMD called “*jmp2ret*, which prevents speculation by replacing returns in the kernel with direct jumps to a return thunk’ (Wikner and Razavi 2022), the purpose of this mitigation is to reduce the surface of the attack target to one single return from all instructions (rather than a return from each instruction) thus lowering the amount of information ‘leakable’ from the returns. While the previous is now implemented by AMD CPUs, Intel have also developed a mitigation proposed by Wikner and Razavi (2022) called eIBRS (enhanced Indirect Branch Restricted Speculation) which is an ‘always on’ version of IBRS the intention of which is to isolate instruction executions, when necessary, rather than all the time. This works

by having the instruction to isolate a particular execution (WRMSR) only running when necessary.

3.0 Methodology

This project has been carried out and a deliverable produced using a variety of tools and sources to ensure a standard of quality is maintained and a level of professionalism remains consistent in every aspect of the project. This section will detail the methodology used in both research and production of the BPG, as well as methods of critical evaluation and review both from the perspective of the project, and from an external third party.

This project has been planned and managed using the Waterfall methodology; the revised model proposed by Winston W Royce (Appendix A), although the model layout is simplified to fit the needs of the project. The waterfall model was chosen as the basis of managing and planning the project as it allows clear stages of planning, implementation, and feedback, while also remaining adaptive for any changes that may occur. Figure 3 is a diagram of the model used in planning and managing the project; it clearly shows sub-sections (those with dashed arrows rather than solid) which, while not absolutely necessary for project completion, do aid it significantly in terms of quality of the final report and BPG. The model created for this project follows the base format, analysis of what is needed, creation of the structure of various deliverables and documents, write-up of the initial document and deliverables, acquisition of feedback from third parties, a changes stage in which feedback is taken into account, and finally the official completion of the project and all deliverables being ready to be deployed/published.

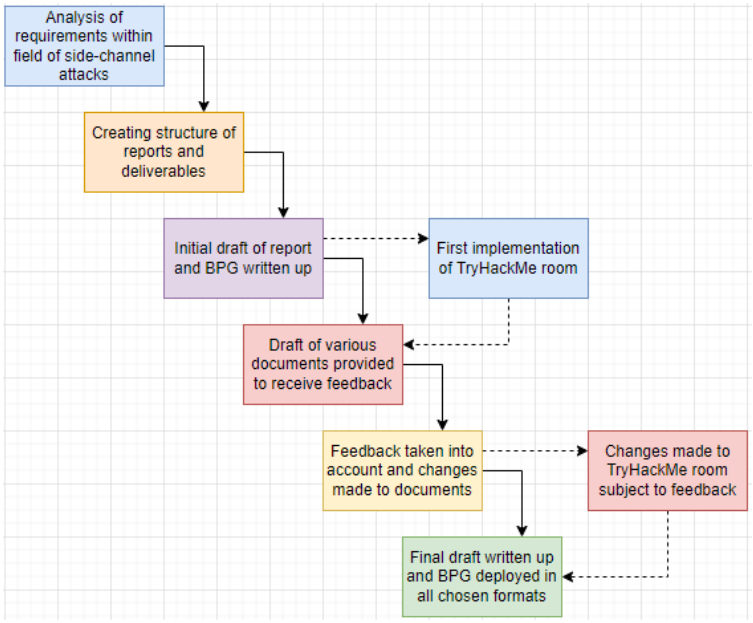


Figure 3 Adapted Waterfall model used in the context of the planning and management of the project

Creating this report following professional standards and guidelines required further reading on the topic of authoring scientific research articles (as this is a primarily research focused article). One source of further reading provided a great deal of information on both the style and substance of various sections of the report, Cargill and O'Connor (2009) provided an effective framework for initially laying out the report (of course some changes were made to better suit the type of research being carried out. The structure provided, which Cargill and O'Connor (2009 p. 9-15) refer to as 'AIMRaD' which is a standard article structure that includes 'Abstract, Introduction, Materials and methods, Results, and Discussion' which has been applied to this report as a starting point. A highly effective graphical representation of the format used in this report is the hourglass structure model created by Cargill and O'Connor (2009 Fig 2.1 p. 10-11).

The 'Study Skills Handbook' by Cottrell (2003) provided a great deal of information on the practice of planning of this project (and led to the use of the Waterfall model as the management plan). Cottrell (2003) has many sections within her book that outline methodology for writing subjectively and providing evidence and professionally reviewed papers to support both statements and questions. One quote that has been a significant influence on the writing style, 'use evidence selectively, too many may obscure your line of reasoning.', this is evident throughout the paper as any references made link well with previous or upcoming statements

and ideas, without taking away from the primary research carried out, and overshadowing the ideas put forward in the title, aims, and objectives of this paper.

To research the overall concept of a CPU-based side-channel attack only professional, peer-reviewed, or journal published sources were used. To ensure only these sources were used, only official repositories of research papers and journals were used, e.g., IEEE. Referencing of these sources followed the USW Harvard format to ensure a standard style is used for ease of reading. The sources will be critically viewed to avoid any bias that may come from personal view of the research or from any bias that may exist in the source. Furthermore, results of each study researched will be viewed objectively as to ensure the results and conclusion of this project being influenced by the opinions or biases of any other researcher.

Understanding the concept of side-channel attacks required personal learning to provide both a comprehensive report, and a concise and understandable guide. To understand the various concepts and ideas within the broad field of side-channel attacks, a significant amount of time was poured into researching the various families of attacks, as well as the more infamous attacks within each family. Special focus was placed on the more expansive topics (cache-based and power-analysis) as with a general understanding of these, this could then be translated into any other attacks. Evidence of this wide-ranging research is evident in the literature review in section 2.0 and then further within section 4.0 where specific attacks are the focus.

To create the BPG which is intended for academic use, a collection of the most effective sources on both CPU-based side-channel attacks as a general topic, and in further depth on the five chosen attack families as outlined in the previous chapter. The layout and structure of the BPG will follow a technical document format, with specific language being used to suit academia and for the purpose of informing those who read it with a theoretical and practical understanding of the content.

Following along the idea of an online guide with an academic use (i.e., used by lecturers, students, and researchers), the creation of a TryHackMe (TryHackMe 2022) room to provide an interactive feature of the guide will allow those who consider themselves kinaesthetic learners to benefit from the use of the BPG whilst also gaining more practical experience not only with the use of side-channel attacks, but also with the debugging in a ‘blue-team’ perspective, i.e. learning how attacks such as these work and how to identify programs that perform them.

To create an effective online teaching and learning tool, it is essential that the complex methods and ideas are translated into a format that can be easily understood for users with little to no prior experience with the topic. This is where the importance of the THM room became apparent, having this sandbox environment where a beginner could learn and watch demonstrations whilst answering questions to cement their understanding of side-channel attacks. The methods and language used to create this effective tool was greatly influenced by Akhmedova (2022) who stated that the creation of interactive teaching materials within computer science and information technology should focus on two central ideas, ‘person centered approach’ and ‘individual and differential approach to students in education’. The first idea of a person-centred approach can be achieved through the creation of materials that appeal to all types of learners by creating both interactive and passive resources. This reason is the primary justification for the use of THM rooms as it allows the publishing of written material, which requires users to read, audio/visual material, which users can see demonstrations, and interactive VMs. All the learning materials provided previously are accompanied by questions which will reinforce the knowledge acquired, ensuring that the interactive materials are accessible to all learners. The second central idea of having an individual and differentiated approach to the audience can be reinforced through a multiplatform guide that allows users to either read a detailed guide online/downloaded offline or take part in the aforementioned interactive materials. The focus on these two ideas is further reinforced by Akhmedova (2022) as they state, ‘it is necessary to develop work plans and theoretical material and adapt the educational process to this approach.’.

Finally, each aspect of the work will be evaluated and critically reviewed to ensure the highest possible standards of work are achieved. To do this the work will be written up in drafts, draft one which will be evaluated and reviewed by the project lead, draft two which will be evaluated and reviewed by a primary supervisor, and then finally a third draft which will be the final, finished product. This will allow both changes felt necessary by the project lead can be implemented and changes in either language or structure can be carried out before a refreshed draft is then sent to the supervisor(s). The supervisor(s) will then provide feedback on more specific details such as content in various chapters or layout/structure of various sections. Finally, the third draft will be the culmination of all changes that may need to be

made after the refreshed draft is sent to supervisor(s). This will be considered the final revised paper.

4.0 Side-Channel Attacks

This section will go into greater detail on five chosen side-channel attacks, these being, *power-analysis attack*, *cache-based*, *time-based*, *remote timing*, and *acoustic cryptanalysis*. The reasoning for selecting these five over others is simply down to the resources available, there are several professional papers available which quite effectively explain each attack, but they are written under the presumption that the reader will already understand the topic at hand. This section will be written only under the assumption that the reader is a competent IT/technology specialist who is capable of understanding concepts, but not methods, of side-channel attacks. For an extremely simple yet concise diagram, refer to appendix S.

4.1 Power-Analysis Attack

Power-analysis attacks are those that use the power output by a device (CPUs in the case of the report) when it is performing certain processes, such as cryptographic processes or operations. In the most basic terms, examining the power output allows the discovery of the secret key used in a cryptographic operation and in turn information that is in use by said device. A study by Ren *et al* (2016) exposed a significant threat from *power-analysis* against 32-bit CPU smart cards, particularly within China where the use of these devices is more prevalent. During their study, they had created a PoC attack based on the *power-analysis* concept and were able to ‘mount successful attacks on the 3DES in a real-life 32-bit CPU smart card’, a successfully attack being that which was able to recover the 3DES keys (the secret keys used for encryption/decryption). A study by Le *et al* (2008) explains, at length, the technical workings of *power-analysis attacks*, along with the two distinct categories:

> ‘Attacks without a reference device (e.g., Differential Power Analysis, Correlation Power Analysis) and attacks using a reference device (e.g., Template Attack, Stochastic Model Attack).’

Le *et al* (2008) explains that attacks without a reference device (as mentioned above) can be derived from the original Differential Power Analysis (DPA) attack, ‘which is based on the fact that the power dissipation to manipulate bit *b* to 1 is

different from the power dissipation to manipulate it to 0' which means that the attack analyses the power used to manipulate the values of various bits, noticing even the slightest difference in power output. DPA attacks also incorporate a concept referred to as Hamming weight which is defined as the number of symbols within a string (in the case of DPA attacks, the number of bits) that differ from the default bit value of 0, which Warren Jr (2002) (See appendix C) also refers to as the 'population count' of a string of bits. Within their paper, Le *et al* (2008) propose an equation to calculate the correct key used in the cryptographic system (Figure 4). Broken down this equation has several distinct parts, Δ is the difference in the key assumption each time a key guess is attempted, *b* refers to the bit that is being manipulated (or not) and whose result relies on the power dissipation occurring. The Σ (sum of) the further fractions refer to the sum of the $G_{0,k}$ and $G_{1,K}$ groups when multiplying them with the DPA signal (which corresponds to the key assumption). Finally, the sums of these are then divided by the total number of elements within the G groups (as denoted by $N_{1,k}$) and the results of both fractions then subtracted from one another, giving the result of the key assumption as

$$\Delta 1, k(b) = \frac{\Sigma G1,kW(Ci)}{N1,k} - \frac{\Sigma G0,kW(Ci)}{N0,k}$$

Figure 4 Le et al (2008) equation to calculate the secret key through a Mono-bit Differential Power Analysis

Le *et al* (2008) also included a graphical representation (Figure 5) of the result of the previous equation showing DPA signals when both correct and incorrect keys are discovered, giving a visual representation of the DPA attack.

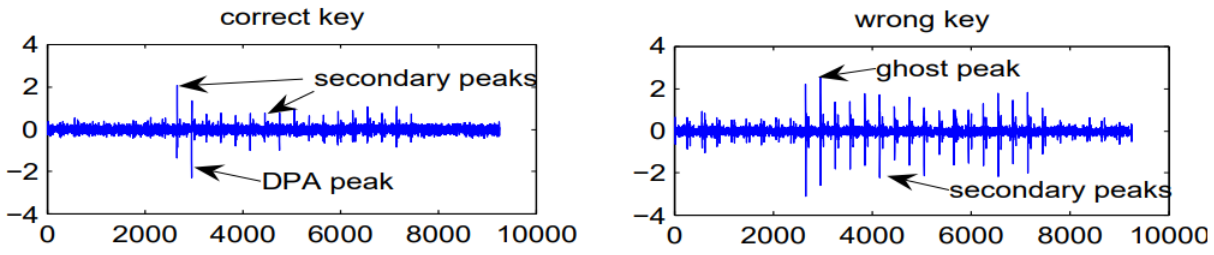


Figure 5 Graphical representation of both correct and incorrect key guesses

Along with the *Power-analysis attacks* that do not use a reference device, there are also those that do. A *template attack* as defined by Le *et al* (2008) as having two stages 'the profiling stage to learn about the device and the key

extraction stage to detect the secret key.'. The *profiling stage* they refer to uses a variety of signals in the creation of a database (this database being dedicated to one singular device) which becomes an integral part of the *template attack* as it stores various templates which are used during the attack. The definition of a 'template' within the bounds of a *template attack* means the number of templates referenceable is extremely large. Le *et al* (2008) provides an example of how large using DES, for which they state there are, '2^6 possibilities for C_i, and 2^6 possibilities for K_k. Hence there are in total 2^{12} = 4096 templates', *C* referring to the text and *K* referring to the key assumption. As the number of templates is so large using a DES system as the basis of the attack is impractical to say the least, for this reason Le *et al* (2008) recommends using the 'bit values like the output of an S-box' as its output contains four bits and as such, 16 templates (2^4). The *key extraction stage* is used to determine the correct key, which uses database that has been created using the templates in the *profiling stage*, to determine the most successful template within the database, the *maximum likelihood metric* (ML) is used which is applied to the result of the extraction of the information, the best template to use is then considered the on with the highest ML. Alongside the ML, Le *et al* (2008) also refers to a 'minimal distance metric' (MD), which is used in conjunction with the ML metric in representing the distributions of templates in a graphical format (which they had displayed in their Figure 11, (see Figure 6 below)).

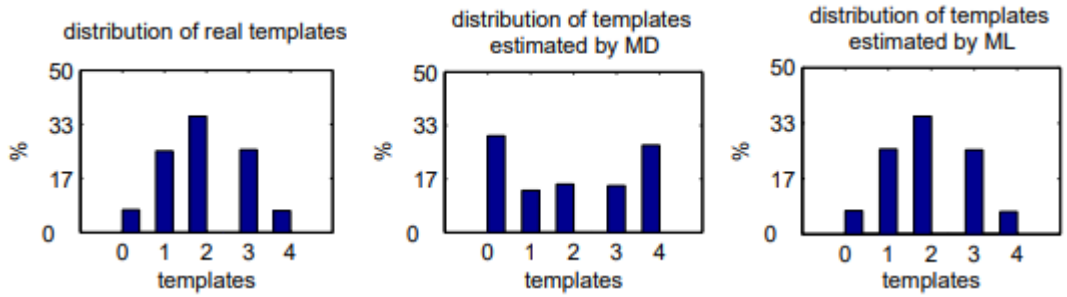


Figure 6 Example used in Le et al (2008) (referred to as Figure 11) with a scenario using a signal size of L = 100. This represents the distribution of templates both in a real sense and using the MD and ML metrics.

While many *Power analysis attacks* require physical access to the CPU/device being targeted, Lipp *et al* (2021) proposes attacks that can be defined as 'software-based power side-channel attacks' which are aimed at various Intel servers, desktop, and laptop CPUs. The attack is performed using the RAPL interface which is Intel's *Running Average Power Limit* as access to this interface does not require privileged access and as such allows direct access to the values

that ‘correlated with power consumption, forming a low-resolution side channel..’ Lipp *et al* (2021) has stated that using ‘sufficient statistical evaluation’ of the values provided by RAPL, they can see clear ‘variations in power consumption, which distinguish different instructions and different Hamming weights of operands and memory loads.’. Furthermore, using this method of attack, Lipp *et al* (2021) shows how ‘an unprivileged attacker is able to leak AES-NI keys from Intel SGX and the Linux kernel’ along with breaking the KASLR (Kernel Address-Space Layout Randomization). SGX refers to the instruction code set used within some Intel CPUs that allow a number of user and operating system level codes to define areas of memory to be defined as private, which are referred to as *enclaves*, the contents of which is intended to be inaccessible from any external source. As a basis for their study, Lipp *et al* (2021) used existing studies on software-based attacks against mobile devices for websites (Qin and Yue 2018) and ‘app fingerprinting, UI inference, password length guessing, and geolocation estimation.’ (Yan *et al* 2015). Lip *et al* (2021) propose PLATYPUS (Power Leakage Attacks: Targeting Your Protected User Secrets) that uses observed changes in ‘power consumption with a resolution of up to 20 kHz’, allowing the execution of various instructions and operands to be differentiated by power usage within the processor. To perform the attacks, Lipp *et al* (2021) repurposed ‘previously published techniques for microarchitectural attacks and apply them in our software-based power analysis attack.’ Such techniques are called *Single-Stepping* and *Zero-Stepping*. *Single-Stepping*, as first introduced by Van Bulck *et al* (2017), which interrupts ‘the victim enclave after *every* single instruction’ by using SGX-Step, an open-source framework for SGX enclaves, to ‘configure untrusted page table entries and/or x86 APIC (Advanced Programmable Interrupt Controller) timer interrupts’ which was created and maintained by Bulck *et al* (2022). *Single-Stepping* works by manipulating the APIC to ‘construct high-resolution, low-noise channels to spy on enclaved execution,’ which allows observations per instruction (also referred to as *‘maximal* temporal resolution’ by Bulck *et al* 2017), Figure 7 by Bulck *et al* (2017) provides an effective graphical model of the framework for performing single-stepping on SGX enclaves. The full process of *Single-Stepping* follows a sequential six step format, step one involves the APIC timer sending an interrupt which then arrives at the enclave, in step two the processor executes the procedure responsible for storing the ‘execution context in the enclave’s SSA (State Save Area) frame, initializes CPU registers, and vectors to

the kernel-level interrupt handler.' which then leads to step three during which the kernel module created by Bulck *et al* (2022) is then registered within the APIC even call back list, allowing it to be called on each occurrence of the timer interrupts. By having this occur on each interrupt, highly customized and specific kernel-level code can be inserted into the enclave's SSA frame, allowing SGX-Step 'to retrieve the stored instruction pointer from the interrupted enclave's SSA frame using the EDBGRD instruction..' This feeds into step four in which the kernel returns to the user space AEP (Asynchronous Exit Pointer) trampoline which is used to release resources of completed processes and executions while waiting for an asynchronous/parallel process to complete. Once ready, the next step, step five, causes 'any attack-specific user mode [made] spy code can be easily run, before the single-stepping adversary configures the APIC timer for the next interrupt, as this step ends the final step, leads to the ERESUME instruction to run, which is defined by FelixCloutier (2019) as an instruction which 'resumes execution of an enclave that was interrupted due to an exception or interrupt, using the machine state previously stored in the SSA.'.

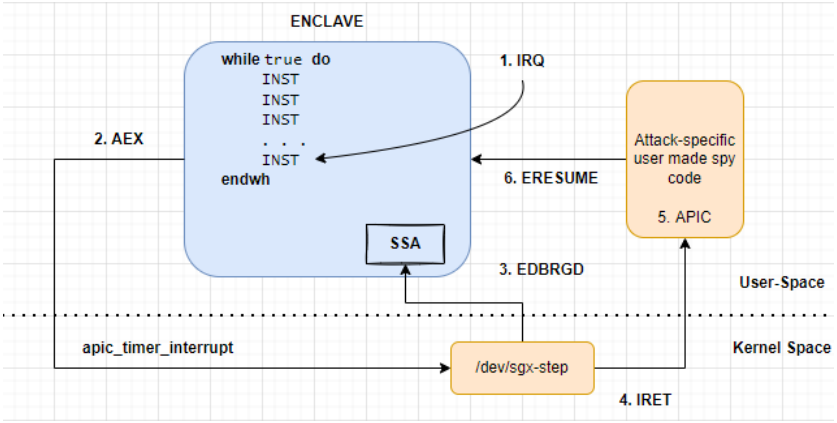


Figure 7 Framework diagram created for this report using the model created and utilised by Bulck et al (2017) (referred to as Figure 1) which outlines the process of Single-Stepping SGX enclaves

Mitigating these *Power analysis attacks* relies on keeping statistical values on power consumption hidden (or at least protected behind an admin privilege level).

4.2 Cache-Based Attack

While many attacks are named for the feature of a CPU or cryptographic device they target, *acoustic*, *timing*, *power*, the *cache-based* attacks are named for the side-channel they take advantage of, namely the cache. Different attacks within this family can use different cache types as side-channels for information leakage but the

most infamous two attacks, Spectre and Meltdown, use the cache memory and memory addresses within the cache. This section of the report will focus solely on Spectre attack as this can be seen as the cause of the growing concern around these forms of attacks.

As previously mentioned in section 2.0 Spectre is a side-channel attack that attacks the *speculative execution* and *branch predictor* of a CPU, which are the components used in improving the performance by queuing predicted instructions and operations based on a predicted (or speculated) result. These queued, predicted instructions can sometimes be mispredicted and then become categorised as *transient instructions* which is at the core of the Spectre vulnerability. Transient instructions are a danger to CPUs as they can affect and modify the microarchitecture of the processor, even when the instruction is reversed, leaving no previous trace of the instruction execution within the architecture of the CPU. Jann Horn and Paul Kocher both discovered the Spectre vulnerability, Horn with Google's Project Zero (2018) and Kocher in collaboration with Daniel Genkin, Mike Hamburg, Mortiz Lipp and Yuval Yarom in June 2017 (although the vulnerability was not made public until January 2018). The presentation by Paul Kocher at the Security and Privacy Symposium (2019) includes a summary of how the Spectre attack is performed, first the speculation scenario (and thus a computational error) is theorised which leads to a normally safe computation (this could even be a simple 'if else' statement) that when speculated, becomes unsafe and causes a computational error with the transient instruction and thus the error required or desired to lead to the information leaking through a side-channel.. A diagram created by Kocher *et al* (2018) effectively shows the theory behind the branch predictor and how it works in practice (Appendix V), this feature of most modern CPUs will speculate that a certain instruction set or function will need to run and prepare this for execution, upon reaching this instruction set, if the speculated instruction was incorrect the result of this is reversed (thus the term transient instruction is then assigned). However, this reversal; as stated previously, does not undo the microarchitectural changes and as such the state of the processor can be obtained by an attacker, providing information leakage through a side-channel (in the case of cache attacks, the side-channel is the cache memory used in programs). Spectre is delivered via a program which an attacker will use to mistrain the branch predictor to get transient instructions to affect the state of the processor's microarchitecture, which then opens the way for

information to leak through the cache memory. Kocher *et al* (2018) states that a branch predictor can be ‘mistrained’ by running the function/call a few times with either the same or similar values that satisfy a certain condition, which leaves the branch predictor open to *predict* and then *speculatively execute* the mispredicted instruction.

An essential idea to note regarding both speculative execution and the branch predictor is that the result (being transient instructions and the Spectre attacks) are not bugs in the sense that there is an error in the computational process of the CPU. A bug is defined as something that happens that is unexpected or something that can be considered outside of the scope of what the device or software should do. All elements involved in Spectre all compile ‘with architecture specs’ (Kocher 2019), the branch predictor learns and predicts based on previous results, the speculative execution correctly reverses the architectural state, the read instructions only fetch information the victim is allowed to read, the caches are storing the state of the microarchitecture, and all covert and side-channels are well known.

Kocher (2019) has proposed the idea that Spectre, rather than being something developed in response to improving security, is actually a ‘symptom of excessive architectural ambiguity’ which he explains is caused by the lack of defined rules that state whether information is shared (or not) between different processes and thus if all processes are given access to the same information, all processes need strengthening.

4.3 Time-Based Attack

A time-based side-channel attack (also called a timing attack) uses the time taken to process different inputs into a system (such as a CPU performing a cryptographic operation), when measuring the time taken to perform particular operations using particular inputs, the ‘timing channels can leak data or keys across a controlled perimeter’ and while it is thought that this information is of minor note, Kocher (1996) presents a number of attacks that ‘exploit the timing measurements from vulnerable systems to find the entire secret key’.

An attacker can analyse these varying processing times with precision instruments, focusing on the processing times of each input. The analysis of each allows an attacker to move backwards through the inputs with the result of compromising the cryptography system in place and thus leakage of information

from the CPU into a side-channel. Kocher (1996) explains that ‘the attack can be tailored to work with virtually any implementation that does not run in fixed time’ and that there is a sole requirement for beginning any of the attacks, calculating x (also referred to as the secret key). Both Diffie-Hellman and RSA can be attacked in a comparable manner as Kocher (1996) explains that both are defined as ‘private-key operations’ which, at their core, ‘consist of computing $R = y^z \bmod n$’. Within this equation, the n is known by all, also referred to as public, y is discoverable by an eavesdropper, after which the attacker’s primary goal is the discovery of the value of x, the secret key. For the attack to succeed, the victim must compute $y^z \bmod n$ over several iterations, with the value of y changing each time. Within these several iterations, the attacker will be privy to y, n, and the computation time (this can be prevented however, if new secret exponents (key) of x are chosen for each iteration). Using the above calculation, any non-fixed time cryptographic operations can be attacked with a timing attack, allowing any attacker to retrieve the secret key x, using it on the cipher text and then retrieving the data that was initially secured, this allows virtually any similar cryptographic operations to be broken by retrieving the secret key all through simply applying known values and the computation time into an equation.

Mitigating timing attacks, as proposed by Kocher (1996), come in both architectural and physical controls, one such architectural mitigation technique can be introduced for ‘blinding signatures’ which prevents an attacker from being privy to the ‘input to the modular exponentiation function.’. A further architectural mitigation is the introduction of ‘constant-time cryptography’ which ensures all operations take a constant amount of time, which relates to the operation that takes the longest. This prevents any relevant information regarding the time taken to perform an operation from being involved as the time remains ‘constant’ throughout the life of the device (ChosenPlaintext 2022). There is a similar (yet far more inefficient) method of mitigation that uses the same concept as constant-time cryptography by suggesting each instruction or operation is followed by a random ‘sleep’ period which then eliminates the precise measurements required for performing such attacks. This is inefficient however as an attacker can just take more measurements to see through the ‘noise’ or random ‘sleep’ periods, which again is countered by adding more noise, leading to a significant slowdown in code execution.

4.4 Differential Fault Analysis Attack

First proposed and used by Biham and Shamir (2006), the Differential Fault Analysis (or DFA) attack that allows an attacker to analyse a number of ciphertexts (anywhere from 50-200) and then 'extract the full DES key from a sealed tamper-resistant DES encryptor'. As stated, DFA is capable of breaking 'many additional secret key cryptosystems including IDEA, RC5, and Feal', which further lends to the danger of this attack. Biham and Shamir (2006) go on to present a model which they call 'asymmetric fault' which can be performed on a cryptographic device (such as a CPU), allowing an attacker to extract the secret key with no knowledge of the 'structure and operation of the cryptosystem'. The base operation of this attack as outlined by Biham and Shamir (2006) is

> 'The smart card is assumed to have random transient faults in its registers, with some small probability of occurrence in each bit, so that during each encryption/decryption there appears a small number of faults (typically one) during the computation, and each such fault inverts the value of one of the bits, either from zero to one or from one to zero'

This is shown in their paper as they describe how an attacker will use the smartcard (or cryptographic device) to encrypt unknown plaintext into ciphertext twice. Upon comparison of the two encryptions if a difference is observed then it can be assumed that a fault occurred in one of the encryptions, providing them with two distinctly different ciphertexts derived from the same plaintext and encryption technique. This attack has been proven to be effective in finding the whole last subkey within the analysis of 50-200 ciphertexts' by 'simulating random single-faults in all the rounds'. The discovery of the subkey is essential as it contains 48/56 of the key bits, the knowledge of which allows an attacker to guess the 'missing 8 bits in all the possible $2^8 = 256$ ways' (Biham and Shamir 2006).

This form of attack can be mitigated easily by increasing the number (generally doubling) 'of encryption operations during a single round' which Khan *et al* (2014) states 'has proven to mitigate such kind of attack' which is further supported by Zhou and Feng (2005) who discovered an effective countermeasure to this type of attack by computing 'the whole or a part of the rounds twice (including key scheduling).' Although this has been noted to 'degrade the whole performance.'

4.5 Acoustic Cryptanalysis Attack

Best defined by Genkin *et al* (2017), ‘the attack can extract full 4096-bit RSA decryption keys from laptop computers (of various models), within an hour, using the sound generated by the computer during the decryption of some chosen ciphertexts’. Genkin *et al* (2017) has shown this type of attack can be performed ‘using a plain mobile phone placed next to the computer, or a more sensitive microphone placed 10 meters away.’, which shows the capability of this attack as it can be performed by relatively low-tech recording devices as long as they are close enough to the victim’s device. To begin this attack, Genkin *et al* (2017) first had to differentiate between the different operations (and accompanying sounds) performed by the CPU, which was done by running a custom program which loops several instructions, these instructions were ‘HLT (CPU sleep), MUL (integer multiplication), FMUL (floating-point multiplication), main memory access (forcing L1 and L2 cache misses), and REP NOP (short-term idle)’. An example of the sound recorded upon execution of each of these operations is available from Genkin *et al* (2017) labelled as Fig. 5 in their paper, for the purpose of completeness it has also been included below as Figure 8.

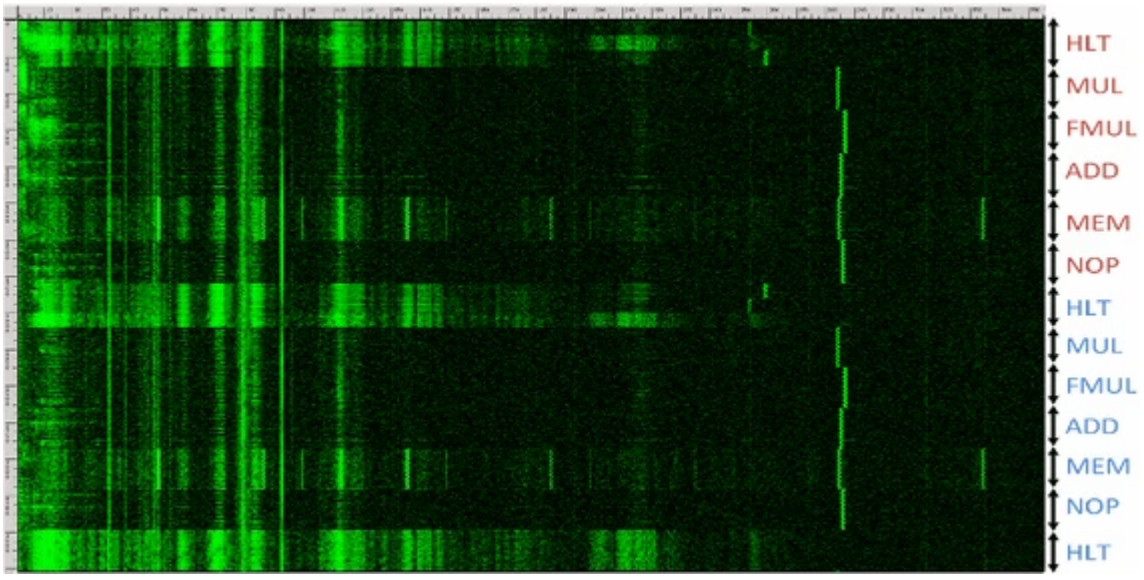


Figure 8 Genkin et al (2017) Fig.5 Acoustic measurements of various CPU operations recorded using a spectrogram

Tromer (2004) specifically targeted the sound produced when using the RSA cryptographic operation. He had chosen the GnuPG 1.2.4 implementation, using it to write short messages and then ‘signing’ them using the 4096-bit RSA encryption key. As with the later paper by Genkin *et al* (2017), Tromer (2004) began the use of a

spectrogram to visualise the sound produced by the CPU, while also elaborating that the bright horizontal strips (which are also visible on Figure 8) represent the computer being in an ‘idle’ state. Alongside this visible idle state, Tromer (2004) also proves the transition between the two exponents (the transition between plain to ciphertext) is clearly visible as ‘halfway through the signing operation there us a transition at several frequency bands’. An key part of this attack is when the attacker is able to distinguish between different keys within RSA, particularly identifying these keys solely on sounds as each key causes its own unique sound, Tromer (2004) himself states that using ‘GnuPG 1.2.4 to sign a fixed message using 7 different 4096-bit RSA keys randomly generated beforehand’ he was able to view a unique spectral signature each time a signing was completed.

The danger with this attack is how easy it is to collect the recordings, as previously mentioned a mobile phone placed near the target device is sufficient in collecting the information required to perform the attack (although it is still complex in utilising this data as a spectrogram would be required to fully understand the data). The danger of this attack can be mitigated in a number of ways, both physical and digital, Tromer (2004) provides a number of possible methods, such as the use of ‘sound dampening equipment’ in an attempt to reduce or eliminate the noise produced by the target device, or on the other end of the same idea, masking the produced sound by performing a louder, more over-bearing sound. Further physical mitigations as suggested by Tromer (2004) is to ensure ‘careful circuit design and high-quality electronic components’ which will reduce the vibrations and therefore sound produced by the device. One digital mitigation proposed by Tromer (2004) is the implementation of further ‘algorithmic techniques to reduce the cryptanalytic usefulness of the emanations to attacker’.

5.0 Best Practice Guide

5.1 Planning and Design

To prepare a foundation for the BPG, Appendix Y was written to both plan the layout and store all information regarding the BPG (structure, content, deadline, and scope). Using this, an effective guide was developed which has a concise section on five different side-channel attacks, those who suggested/proposed such attacks, and the most effective mitigation strategies. Again, as stated previously, this guide is aimed at academics such as lecturers, students, and other researchers who may use this as either a reference, or appendix content to learn more about the topic of side-channel attacks.

The structure of the guide followed a simple format. The first page contains a brief introduction to the content of the guide, followed by a short section on the author alongside a table of contents (hyperlinked for online users). Finishing with a page long section on the topic of side-channel attacks, introducing the reader to the theory behind the attacks, and brief examples of the attack. After planning the structure, a brief of the content was created, describing five side-channel attacks including (Acoustic Cryptanalysis, Timing, Cache, Power analysis, and Allocation based attacks), how they perform in practice, and how they are best mitigated.

The deadline and scope were also considered within the plan. The deadline of the end of July was chosen as August was considered the month for changes to the final draft and finalising both deliverable and report. The scope of the guide required setting some limitations as to the content within, as it is easy to include too much detail and saturate the guide with equations and scientific formulae, particularly on certain attacks. By enforcing a limit of only outlining attacks while providing an in-depth description of the technicalities of each, using peer-reviewed sources, the content of the guide could be considered comprehensive whilst retaining the purpose of providing a resource for academics (students, lecturers, researchers etc.) to use.

5.2 The Guide

The guide itself is in two forms, a physical/online guide book which can be read here, or as a physical copy (only provided on request), and in the form of a ‘TryHackMe’ room which allows a more hands-on approach to the topic which is available here. The guide as a physical or online resource contains a plethora of information and references, allowing those who consider themselves as visual or

aural learners to benefit from it, as well as having the easily accessible (as is it downloadable) file available whenever it is needed. The TryHackMe room allows those who consider themselves to be kinaesthetic learners to answer various questions on the topic, being rewarded with points should they complete the room. A video within both in the TryHackMe room and the appendices of this report (see appendix P) provides a visual demonstration of the Spectre side-channel attack, using a PoC file stored in a repository hosted on GitHub.

5.3 Deliverable Evaluation

The development of the guide has been a significant hurdle in this project as it has been necessary to fully understand each aspect of side-channel attacks (particularly those included within the guide). However, the TryHackMe room has been an extremely effective tool in teaching the basics of at least one side-channel attack, as well as putting forward and teaching the theory behind all side-channel attacks. The idea that information can leak through a side-channel once sufficient information on the system has been attained (such as the secret key, cryptographic operation, or memory addresses used) has been effectively shown using the guide.

While having similar content to the report, the guide provides simpler information which targets academics who may not be as versed about side-channel attacks, whilst also going into detail where appropriate. By compacting the information relevant to the reader, the guide allows a basic understanding which the author feels is more relevant for the target audience. The guide also works very well both standalone and parallel to the TryHackMe room as both contain different topics. While the TryHackMe room focuses on the cache-based side-channel attacks (specifically Spectre), the guide provides overall information on five chosen attacks and their mitigation strategies.

For future development, there are several areas of improvement particularly within the TryHackMe room which suffered due to the time constraints that existed after a sizeable portion of time was allocated to the guide and report write-up. One improvement would be the inclusion of a custom VM (Virtual Machine) in which a user would be able to run the Spectre PoC (or indeed others, such as Meltdown) on a device and see the results first hand. Secondly, more specific attacks within the guide to allow a greater range of information, or more hands-on and casual-user-centric mitigation strategies that those users could attempt without the

need for manipulating the microprocessor or changing specific settings within the CPU. Finally, the guide would benefit massively from a website specifically designed for the topic of side-channel attacks, allowing a repository of sorts to be created. This would house a large number of resources and content that beginners, academics, and CPU/microarchitecture amateurs/professionals can use to both pursue new research and support existing research.

6.0 Conclusion

The purpose of this paper is to provide insight into the concept of side-channel attacks, along with creating a comprehensive guide that provides invaluable information and sources for academics (such as students, lecturers, and researchers) in their own research. The information gathered from a number of primary and secondary sources was standardised, allowing any reader to pick up the paper or guide and understand the content needing only a rudimentary knowledge in the field of IT/computer science.

At the start of this project, there was no one place that had a comprehensive set of information on numerous side-channel attacks and as such the collection of this information (particularly section 2.0 Literature Review) required in-depth understanding of the topic in-order to explain the more complex concepts within side-channel attacks to beginners. This project provided an opportunity not only to learn about professionally researched topics such as *differential power analysis* and *acoustic cryptanalysis* but also to study some more unknown attacks and recent attacks that have been discovered in the last year, such as *Hertzbleed* and *BranchScope*.

Before beginning this project, the author had only knowledge of side-channel attacks through lessons taught within university and as such focused solely on Spectre and Meltdown to meet the requirements of assessments. By using side-channel attacks as the topic for this dissertation, a wider range of knowledge was acquired, along with a greater understanding of the concepts and ideas that are involved in side-channel attacks. One key idea learned through the completion of this project was the use of a public and private exponent in carrying out cryptographic sequences, particularly the involvement of said exponents within side-channel attacks (such as power analysis and timing attacks) as an attack vector. A more specific pair of concepts learned about during this project was the *hamming weight* and *hamming distance* which are defined as a comparison of two bytes, looking at the number of symbols different from zero (0) and the number of differentiating symbols, respectively.

This work has provided a key contribution to the academic sector on the topic of side-channel attacks through the deliverables created. The guide allows access to a peer-reviewed and professional repository of information that works as a ‘getting started’ guide, or as a further reading area. The TryHackMe room provides a secure

location users can test themselves on side-channel attacks and view a demonstration on the Spectre side-channel attack. The guide has also been hosted on a blog created by the author and hosted by Github Pages which allows free hosting of a site, this allows the guide to be available online, and in a downloadable format.

There are a number of areas that can be further developed to improve this project significantly in the future. One such area being the enhancement of the TryHackMe room and possibly even expanding this into its own learning pathway on TryHackMe (a collection of rooms that provide a user with all tools and knowledge necessary to be considered knowledgeable in that area). Another area for improvement would be creating a platform for the learning materials to be hosted on, for example, an expanded version of the website which also hosts tools, source code, and demonstration videos. Finally, expanding the range of demonstrations available on the YouTube channel to cover all types of side-channel attacks would benefit both the purpose of this project and the research area.

With further time and resources, one avenue that would be explored are the practical side of attacks mentioned in section 4.0 Side-Channel Attacks. This includes accessing sensitive sound equipment and spectrographs to record various outputs of the target devices (such as CPUs). This would provide an opportunity to explore the practical elements of these attacks and the technicalities involved allowing the creation of more demonstrations and guides. Finally, this project has raised a number of issues with regards to the standard of writing for many research papers that come under the topic of side-channel attacks. Many of the concepts within, require mathematical equations and an understanding of electronics to fully appreciate them. This project did attempt to solve this by standardising a great deal of information into a less complex style, with further explanations provided where necessary. While successful, collaboration with existing specialists who have developed their own projects and frameworks that look at this specific security vulnerability, such as Mitre's ATT&CK framework (2022), the quality of the language standardisation, and the number of attacks and/or mitigation techniques would improve vastly.

This project has been successful in answering the issue posed by the title 'Avoiding, Preventing, and Mitigating Side-Channel Attacks: A Best Practice Guide' as a comprehensive guide has been created with standard language and

explanations of complex topics alongside a number of distribution techniques that allow users to access this guide in both an online and offline format. Furthermore, the inclusion of visual and interactive materials means those with different learning styles can still benefit. The language used and the format followed make both this paper and the guide suitable for academics, such as students, lecturers, and researchers.

References

Akhmedova, K. (2022) ‘Theoretical Foundations for the Creation of Electronic Interactive Educational and Programming in the Topic “Computer Science and Information Technology”’ *International Journal of Innovations in Engineering Research and Technology* 2022, Vol 9, Issn 4, pp. 150-152, Available at: https://media.neliti.com/media/publications/429596-theoretical-foundations-for-the-creation-b88e707e.pdf (Accessed: 03/08/2022)

AMD (2018) *Software Techniques for Managing Speculation on AMD Processors* developer.amd.com, 01/02, Available at: http://developer.amd.com/wordpress/media/2013/12/Managing-Speculation-on-AMD-Processors.pdf (Accessed: 27/06/2022); archived at *Wayback Machine* (https://archive.org/web/) > http://developer.amd.com/wordpress/media/2013/12/Managing-Speculation-on-AMD-Processors.pdf

Angel, S. Kannan, S. Ratliff, Z. (2020) ‘Private resource allocators and their applications’ *IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 372-391, doi: 10.1109/SP40000.2020.00065.

Azana, D, L. (2017) *Differences between ASLR, KASLR and KARL*, Daniloaz.com, 07/07, Available at: https://www.daniloaz.com/en/differences-between-aslr-kaslr-and-karl/ (Accessed: 20/06/2022)

Biham, E. Shamir, A. (2006) ‘Differential Fault Analysis of Secret Key Cryptosystems’. *Crypto 1997: Advances in Cryptology — Crypto ‘97*, pp. 513-525, Available at: https://link.springer.com/content/pdf/10.1007/BFb0052259.pdf (Accessed: 10/08/2022)

Bulck, J, V. Lipp, M. Gyselnck, J. Bognar, M. (2022) *SGX-Step* [GitHub] Available at: https://github.com/jovanbulck/sgx-step (Accessed: 08/07/2022)

Bulck, J, V. Piessens, F. Strackx, R. (2017) ‘SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control’, *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, Shanghai China, 28 October, Association for Computing Machinery, New York: Association for Computing Machinery, Available at: https://dl.acm.org/doi/pdf/10.1145/3152701.3152706 (Accessed: 05/07/2022)

ChosenPlaintext (2022) *A Beginner’s Guide to Constant-Time Cryptography* [Online], Available at: https://www.chosenplaintext.ca/articles/beginners-guide-constant-time-cryptography.html (Accessed: 08/08/2022)

Canella, C. Bulck, J, V. Schwarz, M. Lipp, M. von Berg, B. Ortner, P. Piessens, F. Evtyushkin, D. Gruss, D. (2019) ‘A Systematic Evaluation of Transient Execution Attacks and Defenses’ *USENIX Security Symposium* 2019, Available at: https://arxiv.org/abs/1811.05441 (Accessed: 29/06/2022)

Cargill, M. O’Connor, P. (2009) *Writing Scientific Research Articles*, 2nd Edn, Sussex: Wiley-Blackwell

Choi, Y. Knies, A, D. Gerke, L. Ngai, T-F. (2001) ‘The Impact of If-Conversion and Branch Prediction on Program Execution on the Intel Itanium Processor’ *Proceedings of the 34th Annual International Symposium on Microarchitecture* 2001, Austin, Texas, USA, December 1-5 2001, Available at: https://www.researchgate.net/publication/221005549_The_impact_of_if-conversion_and_branch_prediction_on_program_execution_on_the_Intel_Itanium_processor (Accessed: 02/08/2022)

Chromium (2018) *Site Isolation*. Available at: https://www.chromium.org/Home/chromium-security/site-isolation/ (Accessed: 27/06/2022)

Cottrell, S. (2003) *The Study Skills Handbook*, 2nd Edn, New York: Palgrave MacMillan

Evtyushkin, D. Riley, R. Abu-Ghazaleh, N. Ponomarev, D. (2018) ‘BranchScope: A New Side-Channel Attack on Directional Branch Predictor’ *ACM SIGPLAN Notices 2018*, [Online], Vol 53, Issn 2, pp. 693-707, Available at: https://dl.acm.org/doi/abs/10.1145/3296957.3173204 (Accessed: 02/07/2022)

FelixCloutier (2019) *ERESUME — Re-Enters an Enclave* [Online] Available at: https://www.felixcloutier.com/x86/eresume (Accessed: 05/07/2022)

Guo, Y. Chen, X. Mei, H. Yan, L. (2015) ‘A Study on Power Side Channels on Mobile Devices’ *Internetware ‘15: Proceeding of the 7th Asia-Pacifica Symposium on Internetware*, Wuhan China, 6 November. New York: Association for Computing Machinery. Available at: https://dl.acm.org/doi/proceedings/10.1145/2875913 (Accessed: 05/07/2022) https://www.researchgate.net/publication/309024581_A_Study_on_Power_Side_Channels_on_Mobile_Devices,

IEEE Symposium on Security and Privacy (2019) *Spectre Attacks Exploiting Speculative Execution*. Available at: https://www.youtube.com/watch?v=zOvBHxMjNls&t=664s (Accessed: 04/08/2022)

Intel (2013) *Enhanced Security Features for Confidential Computing* Intel Platform Security Technologies, Available at: https://cdrdv2.intel.com/v1/dl/getContent/723693?explicitVersion=true (Accessed: 05/07/2022)

Intel (2018) *Speculative Execution Side Channel Mitigations*, Revision 3.0, Available at: https://www.intel.com/content/dam/develop/external/us/en/documents/336996-speculative-execution-side-channel-mitigations.pdf (Accessed: 27/06/2022)

Kim, T. Shin, Y. (2022) ‘ThermalBleed: A Practical Thermal Side-Channel Attack’ *IEEE Access*, vol 10, pp. 25718-25731., doi: 10.1109/ACCESS.2022.3156596.

Kocher, P. (1996) *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, pp. 3-9, Available at: https://paulkocher.com/doc/TimingAttacks.pdf (Accessed: 08/08/2022)

Kocher, P (2019) 'Spectre Attacks: Exploiting Speculative Execution' *40th IEEE Symposium on Security and Privacy* 2019, Available at: https://www.ieee-security.org/TC/SP2019/ (Accessed: 04/08/2022)

Kocher, P. Horn, J. Fogh, A. Genkin, D. Gruss, D. Haas, W. Hamburg, M. Lipp, M. Mangard, S. Prescher, T. Schwarz, M. Yuval, Y. (2018) *Spectre Attacks: Exploiting Speculative Execution*, Available at: https://spectreattack.com/spectre.pdf (Accessed: 27/06/2022)

Kocher, P. Jaffe, J. Jun, B. (1999) 'Differential Power Analysis' *Advances in Cryptology — CRYPTO' 99*, Vol 1666, Available at: https://paulkocher.com/doc/DifferentialPowerAnalysis.pdf (Accessed: 19/06/2022)

Le, T-H. Canovas, C. Clédière. (2008) *An Overview of Side Channel Analysis Attacks*, Available at: https://dl.acm.org/doi/pdf/10.1145/1368310.1368319 (Accessed: 30/06/2022)

Lipp, M. Kogler, A. Oswald, D. Schwarz, M. Easdon, C. Canella, C. Gruss, D. (2021) 'PLATYPUS: Software-based Power Side-Channel Attacks on x86' *2021 IEEE Symposium on Security and Privacy (SP)*, Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9519416 (Accessed: 05/07/2022)

Lipp, M. Schwarz, M. Gruss, D. Prescher, T. Haas, W. Fogh, A. Horn, J. Mangard, S. Kocher, P. Genkin, D. Yarom, Y. Hamburg, M. (2018) *Meltdown: Reading Kernel Memory from User Space*, Available at: https://meltdownattack.com/meltdown.pdf (Accessed: 27/06/2022)

Microsoft Edge Team (2018) *Mitigating speculative execution side-channel attacks in Microsoft Edge and Internet Explorer*, Available at: https://blogs.windows.com/msedgedev/2018/01/03/speculative-execution-mitigations-microsoft-edge-internet-explorer/ (Accessed: 27/06/2022)

Mitre (2022) *ATT&CK Framework*, Available at: https://attack.mitre.org/matrices/enterprise/ (Accessed: 02/09/2022)

Percival, C. (2005) 'Cache Missing for Fun and Profit', *BSDCan 2005*, Ottawa 13 May. Massachusetts Institute of Technology. Available at: http://css.csail.mit.edu/6.858/2014/readings/ht-cache.pdf (Accessed: 16/06/2022)

Qin, Y. Yue, C. (2018) 'Website Fingerprinting by Power Estimation Based Side-Channel Attacks on Android 7' *TrustCom/BigDataSE* 2018, Available at: https://inside.mines.edu/~chuanyue/papers/TrustCom2018Power.pdf (Accessed: 05/07/2022)

Quinn, R. (2021) 'What is "Retpoline?"' *Assured Information Security* Available at: https://www.griffissinstitute.org/what-we-do/gi-lecture-education-series-presentations/giles-retpoline#:~:text=Retpoline%20stands%20for%20return%20and,jmp%E2%80%9D%20and%20%E2%80%9Ccall%E2%80%9D. (Accessed: 22/08/2022)

Ren, Y. Wu, L. 'Power Analysis Attacks on Wireless Sensor Nodes using CPU Smart Cards' *22nd Wireless and Optical Communication Conference* 2013, pp. 665-670, Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6676458&tag=1 (Accessed: 29/06/2022)

Su, C. Zeng, Q. (2021) 'Survey of CPU Cache-Based Side-Channel Attacks: Systematic Analysis, Security Models, and Countermeasures'. *Security and Communication Networks*, Vol 2021, p.1-15. Available at: https://doi.org/10.1155/2021/5559552. (Accessed: 16/06/2022)

Tromer, E. Shamir, A. (2007) *Hardware-Based Cryptanalysis*. PhD thesis. Scientific Council of the Weizmann Institute of Science. Available at: https://www.cs.tau.ac.il/~tromer/papers/tromer-phd.pdf (Accessed: 26/06/2022)

TryHackMe (2022) *Hands-on Cyber Security Training with Real-World Scenarios* Available at: https://tryhackme.com (Accessed: 29/06/2022)

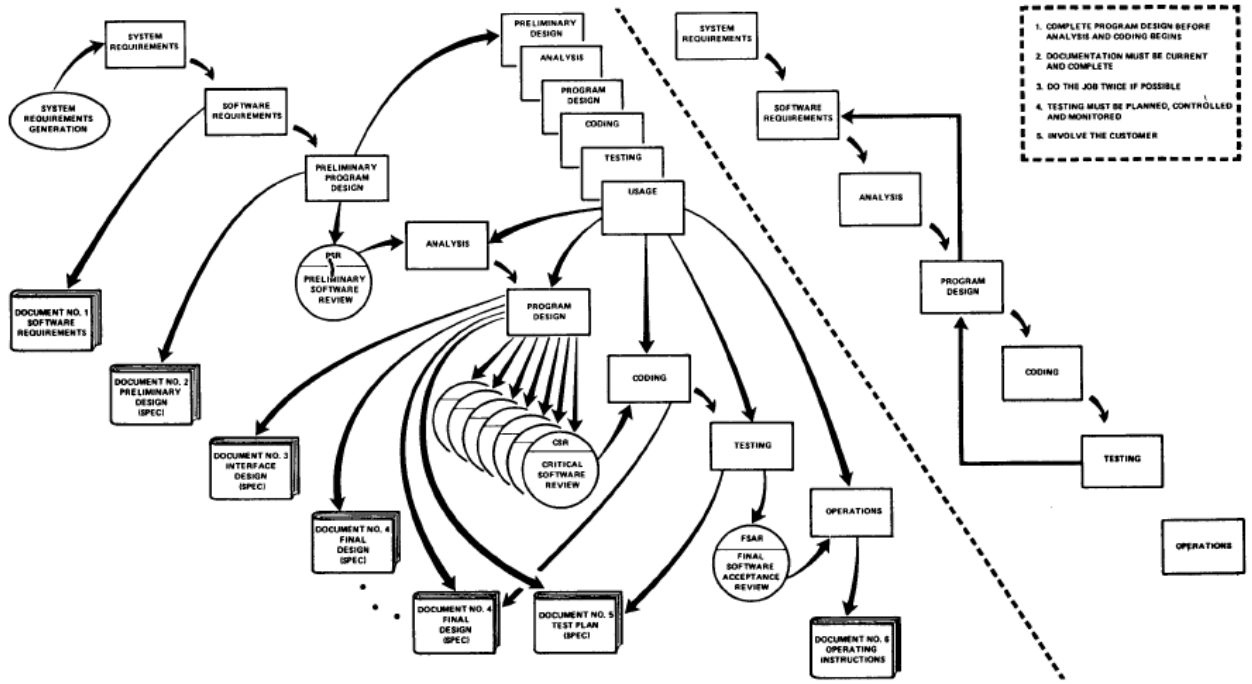
Wang, Y. Paccagnella, R. Tang, H, E. Shacham, Hovav. Fletcher, C, W. Kohlbrenner, D. (2022) 'Hertzbleed: Turning Power Side-Channel Attacks into Remote Timing Attacks on x86' *31st USENIX Security Symposium*, Boston 10-12 August 2022. Available at: https://www.hertzbleed.com/ (Accessed: 16/06/2022)

Wikner, J. Razavi, K. (2022) 'RetBleed: Arbitrary Speculative Code Execution with Return Instructions' *Usenix Security '22 Technical Sessions*, Available at: https://www.usenix.org/conference/usenixsecurity22/technical-sessions (Accessed: 16/06/2022)

Wolaver, D, H. (1991), *Phase-Locked Loop Circuit Design*, Englewood Cliffs, N.J: Prentice Hall.

Appendices

Appendix A



Royce, W, W. (1970) ‘Managing the Development of Large Software Systems’ *Technical Papers of Western Electronic Show and Convention (WesCon)* August 25–28, 1970, Los Angeles, USA. Available at: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf (Accessed: 29/06/2022)

Appendix B

In One Lesson (2013) *How a CPU Works* Available at: https://www.youtube.com/watch?v=cNN_tTXABUA (Accessed: 30/06/2022)

Appendix C

Warren Jr, H, S. (2013) *Hacker’s Delight* 2nd edn Boston: Addison-Wesley

Appendix D

Luo, C. Fei, Y. Luo, P. Muhkerjee, S. Kaeli, D. (2012) ‘Side-Channel Power Analysis of a GPU AES Implementation’ *33rd IEEE International Conference on Computer Design: VLSI in Computers and Processors*, (ICCD) 2015, Available at: https://ieeexplore.ieee.org/abstract/document/7357115 (Accessed: 30/06/2022)

Appendix E

Ambrose, J, A. Ragel, R, G. Parameswaran, S. ‘RIJID: Random Code Injection to Mask Power Analysis Based Side Channel Attacks’ *44th Annual Design Automation Conference 2007* Available at: https://dl.acm.org/doi/abs/10.1145/1278480.1278606 (Accessed: 30/06/2022)

Appendix F

Zhao, X. Guo, S. Zhang, F. Wang, T. Shi, Z. Liu, H. Ji, K. Huang, J. (2013) ‘Efficient Hamming Weight-Based Side-Channel Cube Attacks on PRESENT’ *Journal of Systems and Software* Vol 86 (Issn 3) pp. 728-743 Available at: https://www.sciencedirect.com/science/article/pii/S0164121212003081 (Accessed: 30/06/2022)

Appendix G

Shamir, A. Tromer, E. Genkin, D. (2016) ‘Acoustic Cryptanalysis’ *Journal of Cryptology*, Vol 30 pp. 392-443, Available at: https://link.springer.com/article/10.1007/s00145-015-9224-2#Sec44 (Accessed: 20/07/2022)

Appendix H

Kocher, P (2018) *spectre_poc.c* [GitHub] Available at: https://gist.github.com/anonymous/99a72c9c1003f8ae0707b4927ec1bd8a (Accessed: 11/07/2022)

Appendix I

Schwarzl, M. Borrello, P. Saileshwar, G. Müller, H. Schwarz, M. Gruss, D. (2021) *Practical Timing Side Channel Attacks on Memory Compression,* Available at: https://arxiv.org/abs/2111.08404 (Accessed: 21/07/2022)

Appendix J

Wong, W, H. *Timing Attacks on RSA: Revealing Your Secrets through the Fourth Dimension*. Available at: https://www.cs.sjsu.edu/faculty/stamp/students/article.html (Accessed: 21/07/2022)

Appendix K

Kocher, P, C. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Available at: https://www.rambus.com/wp-content/uploads/2015/08/TimingAttacks.pdf (Accessed: 21/07/2022)

Appendix L

Mushtaq, M. Mukhtar, M, A. Lapotre, V. Khurram, B, M. Gogniat, G. 'Winter is here! A decade of cache-based side-channel attacks, detection & mitigation for RSA' *Information Systems* (2020) Available at: https://hal.archives-ouvertes.fr/hal-02537540/document DOI: https://doi.org/10.1016/j.is.2020.101524 (Accessed: 26/07/2022)

Appendix M

Liu, F. Yarom, Y. Ge, Q. Heiser, G. Lee, R, B. (2015) 'Last-Level Cache Side-Channel Attacks are Practical' *2015 IEEE Symposium on Security and Privacy* 17-21 May 2015, San Jose USA. Available at: https://ieeexplore.ieee.org/document/7163050 (Accessed: 28/07/2022)

Appendix N

Page, D. (2003) 'Defending against cache-based side-channel attacks' *Information Security Technical Report* March 2003. Vol 8 Issn 1, pg. 30-44. Available at: https://www.sciencedirect.com/science/article/pii/S1363412703001043 (Accessed: 28/07/2022)

Appendix O

Black, E, P. (2006) 'Hamming distance' *Dictionary of Algorithms and Data Structures* Available at: https://www.nist.gov/dads/HTML/HammingDistance.html (Accessed: 28/07/2022)

Appendix P

Harris, T (2022) *Demonstration of Basic Spectre POC* Available at: https://youtu.be/UzKwc-MbEVM (Accessed: 15/07/2022)

Appendix Q

Angel, S. Kannan, S. Ratliff, Z. (2020) 'Private Resource Allocators and their Applications' *2020 IEEE Symposium on Security and Privacy (SP)*, Available at: https://eprint.iacr.org/2020/287.pdf (Accessed: 01/08/2022)

Appendix R

Barker, E. (2020) 'Guideline for Using Cryptographic Standards in the Federal Government' *Cryptographic Mechanisms*, Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175Br1.pdf (Accessed: 01/08/2022)

Appendix S - **Standard fundamentals of any side-channel attack**

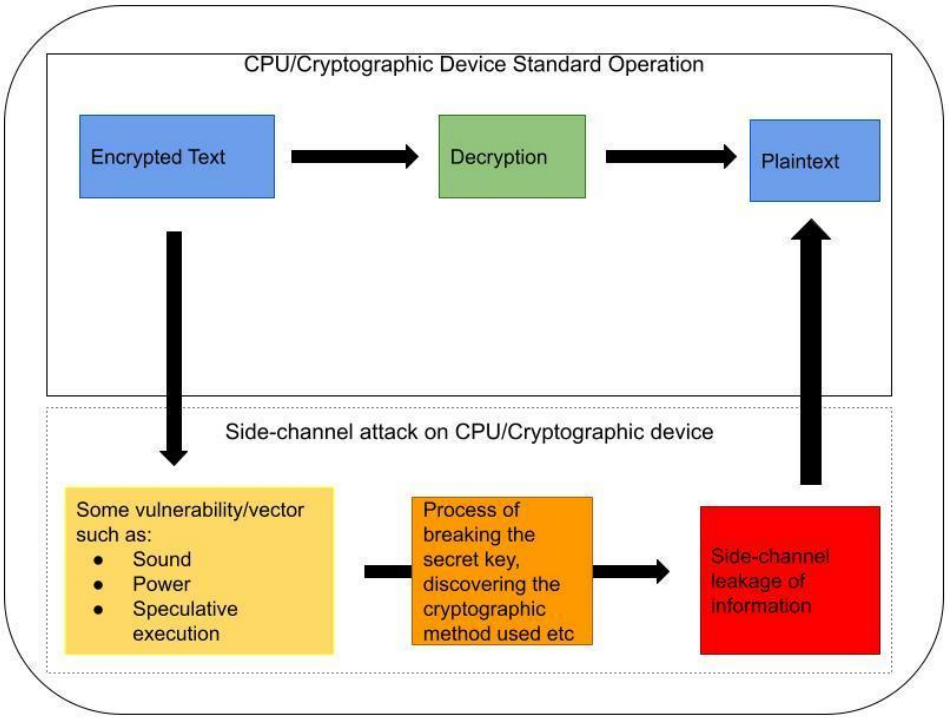
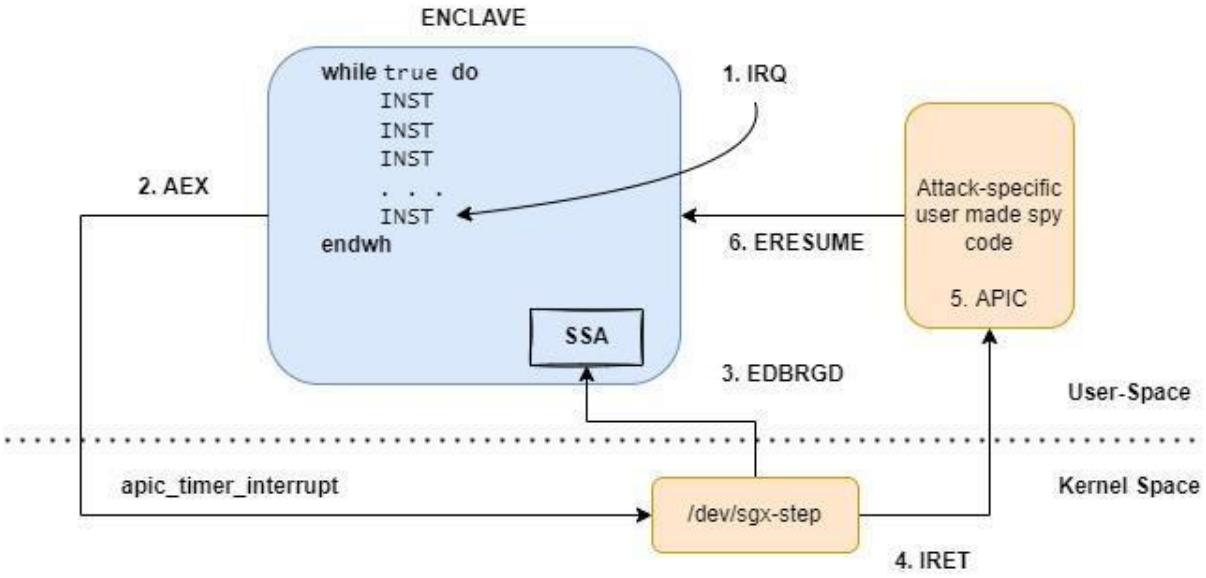


Figure 9 Created for the purpose of this report

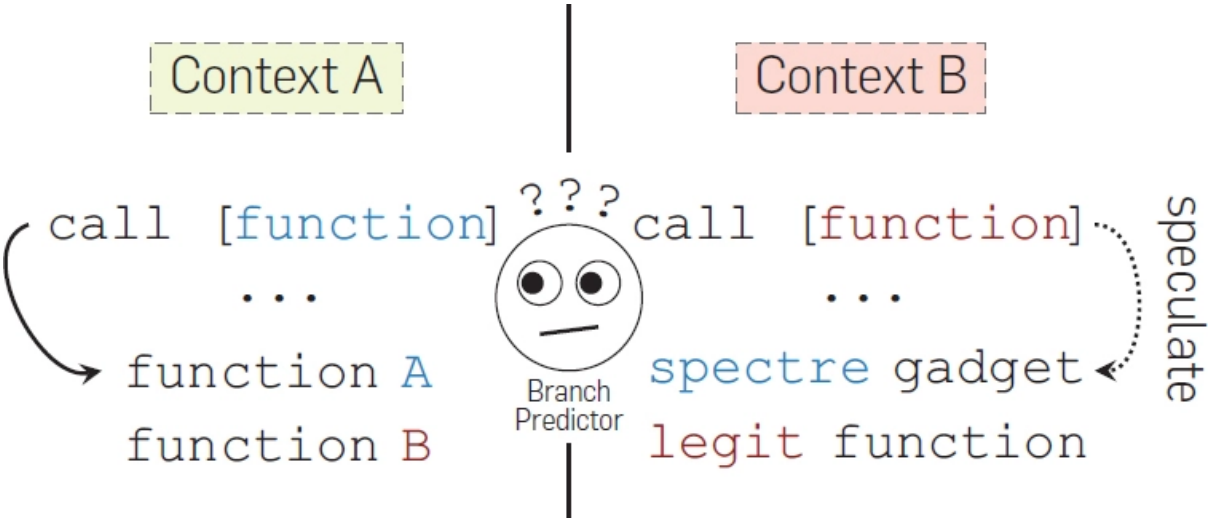
Appendix T - **Diagram of attack on an SGX-Enclave**



Appendix U

Richards, J (2021) *An Integrated Cyber Threat Hunting Program Applying Machine Learning for Enhanced Intelligence Capabilities*, [Online] Available at: https://pure.southwales.ac.uk/ws/portalfiles/portal/9872387/MSc_Thesis_Joshua_Richards.pdf (Accessed: 03/08/2022)

Appendix V - **Concept behind the branch predictor and speculative execution**



Kocher, P. Horn, J. Fogh, A. Genkin, D. Gruss, D. Haas, W. Hamburg, M. Lipp, M. Mangard, S. Preacher, T. Schwarz, M. Yarom, Y. (2020) 'Spectre Attacks: Exploiting

Speculative Execution' *Communications of the ACM* July 2020, Vol 63, Issn 7, Available at: https://cacm.acm.org/magazines/2020/7/245682-spectre-attacks/fulltext (Accessed: 04/08/2022)

Appendix W

Dusart, P. Letourneux, G. Vivolo, O. (2008) 'Differential Fault Analysis on A.E.S' *ACNS 2003: Applied Cryptography and Network Security*, pp 293-306, Available at: https://link.springer.com/content/pdf/10.1007/978-3-540-45203-4_23.pdf (Accessed: 10/08/2022)

Appendix X - **Project Logbook**

Event	Description	Date
Meeting 1	First meeting with Dr. Richard Ward. See 'Meetings' section for information on points discussed - Primary point was the possible topic area and work required.	14/04/22
Meeting 2	See 'Meetings' section for information on points discussed - Primary point was the research proposal and its contents	29/04/22
Research Proposal	Research proposal that contained preliminary information on the project, as well as the outline of any possible ethical issues. This proposal also included an outline and approximation of the time commitments each part of the project may require, alongside a Gantt chart.	09/05/22 - 15/05/22
Baseline research collected	A variety of secondary sources were collected from books, and peer reviewed articles and journals to form a baseline for both the literature review, and initial research section of the report.	25/05/22 - 31/05/22
Meeting 3	See 'Meetings' section for information on points discussed - Primary point was the completion of the Ethics form, which was completed and sent to Richard (the primary supervisor) to be signed. Once the Ethics form was signed, it was then sent to Daniel Cunliffe to be recorded as completed.	06/06/22
Official start of project	After meeting 3, work had officially begun on the project. The structure, format, and layout of the	06/06/222

	document was completed, leaving only the content to carry out. Various comments were left on each section of the structure to ensure each section was understood as it was reached.	
Meeting 4	See ‘Meetings’ section for information on points discussed - Primary point was how to create the BPG, also discussed was how to create a true methodology, primary supervisor also loaned several books on creating and using a good methodology. A fifth meeting was scheduled for June 21st which would be the first in-person meeting of the timeline.	17/06/22
Literature Review begun	Parallel to meeting 4, the literature review was started, and several sources cited and explained regarding the basics of what a side-channel attack is, as well as the various families of attacks that will be analysed and included in the BPG.	17/06/22
Meeting 5	See ‘Meetings’ section for information on points discussed - The purpose of this meeting was primarily for the primary supervisor to provide the books to assist with the writing of the methodology of this project. There was also some discussion on the BPG such as the format (which was decided to be a technical document), and avenues of distributing it. One avenue that was discussed at length was hosting it on TryHackMe as a room (either informational or interactive) to allow users to test a side-channel attack (most likely Spectre) in a sandboxed and isolated environment. There was some minor discussion regarding making the BPG available on the web both as a downloadable attachment and as a section on the website itself (the website would be the one created as part of the bachelor’s dissertation in 2021).	21/06/22
Best Practice Guide Plan	A plan of the BPG has been created and topics/content has been considered. A technical document/manual template was chosen as this allowed professionalism while still being clear and concise.	21/06/22
Best Practice Guide	The best practice guide has been started although formatting will undoubtedly change as it will require fine-tuning to fit the standard of a technical document.	27/06/22
Meeting 6	See ‘Meetings’ section for information on points discussed - The purpose of this meeting was	05/07/22

	primarily to answer queries on the format of the dissertation, particularly about the references, appendices, and figures (and table of). Alongside this, there was some discussion on the current aims and objectives and the primary supervisor had suggested a slight change to Aim 2 from creating a report to selecting the five appropriate attacks. Objectives for Aim 2 and 3 were also changed to better reflect the requirements and more effectively justify the scope of the project.	
Meeting 7	See ‘Meetings’ section for information on points discussed - The purpose of this meeting was a cursory discussion on the practical and deliverable element of the project, the main focus for the rest of July will now be the deliverable, particularly the BPG and the TryHackMe room. A further meeting was arranged for the next week at a different time than usual.	12/07/22
Meeting 8	See ‘Meetings’ section for information on points discussed - The purpose of this meeting was regarding the deliverable progress being made, specifically the creation of the TryHackMe room. Supervisor feedback was positive, and the supervisor also stated that he would run through the room before the next meeting (21/07/2022) to prepare feedback. Upon next meeting a large portion of the guide should be complete and ready for preliminary feedback.	19/07/22
Completion of BPG and request for feedback	The deliverable has now been completed and the supervisor has been contacted for feedback. The deliverable has taken two forms, one is the BPG (Best Practice Guide) which is a detailed guide on five distinct side-channel attacks, along with their mitigation techniques. The second deliverable form is the TryHackMe (THM) room which contains information and an interactive activity which culminates in questions that a user will answer to affirm their knowledge after reading through the resources provided. Finally, within the THM room a video has been created which allows users to watch a live demonstration of Spectre on the creators PC, seeing the leaked information being received through a side-channel.	02/08/2022
Meeting 9	See ‘Meetings’ section for information on points discussed - This meeting was more impromptu than previous as time constraints forced an earlier meeting than intended although this was no issue,	03/08/2022

	both for the project or for the supervisor. This meeting was a short progress update for the supervisor, informing him that the deliverable has been finished, along with most of the report. The supervisor advised adding some more materials to the deliverable (the BPG) such as visual materials describing some of the attacks.	
Plan for visual materials for deliverable	A brief plan and brainstorming document have been put together to create some visual materials for various side-channel attacks. These visual materials will be in the form of diagrams and a video (if time constraints allow).	04/08/2022
Creation diagrams for the report	Alongside the textual content of the guide, a number of diagrams have been drawn up that show the operation of side-channel attacks.	04/08/2022
Website prepared and adapted for deliverable hosting	Originally a personal blogging site created some months ago, the GitHub pages site, tomisee.github.io, will have the guide available both as a download or as a readable article on the site. The site has been created using a one file delivery system using Markdown to easily create a simple location to host this deliverable for free.	09/08/2022