uniserver

D7.2 Countermeasures to Security Risks due to Extended Margins

Contract number	688540
Project website	http://www.uniserver2020.eu
Contractual deadline	Project Month 21: 31st Oct 2017
Actual Delivery Date	Dec 2017
Dissemination level	Public
Report Version	1.1
Main Authors	Philip Hodgers (QUB), Georgios Karakonstantis (QUB)
Reviewers	Yanos Sazeides (UCY), Dimitris Gizopoulos (UoA)
Keywords	Security, Micro-Server, Edge Computing

Notice: The research leading to these results has received funding from the European Community's Horizon 2020 Programme for Research and Technical development under grant agreement no. 688540.

Disclaimer
This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement Nr 688540. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the project and to the extent foreseen in such agreements.

Acknowledgements
The work presented in this document has been conducted in the context of the EU Horizon 2020. UniServer is a 36-month project that started on February 1st, 2016 and is funded by the European Commission. The partners in the project are:

The Queen's University of Belfast (QUB)
The University of Cyprus (UCY)
The University of Athens (UoA)
Applied Micro Circuits Corporation Deutschland Gmbh (APM)
ARM Holdings UK (ARM)
IBM Ireland Limited (IBM)
University of Thessaly (UTH)
WorldSensing (WSE)
Meritorious Audit Limited (MER)
Sparsity (SPA)

More information
Public UniServer reports and other information pertaining to the project are available through the UniServer public Web site under http://www.uniserver2020.eu.

Change Log

Version	Description of change
0.5	Updated draft
1.0	Incorporation of Review Comments
1.1	Added section on Out-of-Order Execution attacks

Table of Contents

Index of Figures

Executive Summary

This report forms part of the cross-layer secure system integration and evaluation activities of UniServer WP7 and examines primarily the security risks associated with the move from a cloud deployment model to the edge computing model implicit within the UniServer project. In contrast to a centralised cloud data centre, edge deployments will be constituted from many small clusters or individual installations, where elevated levels of physical security are not economically viable. Physical security of the micro-server may consist primarily of a light-weight enclosure and, from a security perspective, it should be assumed that a determined attacker will be able to gain full access to the system. This creates a larger threat surface, which now incorporates physical attacks, posing threats to the micro-server and the wider network it connects to. Deployments at the edge should be made under the assumption that networks are operating over untrustworthy links, with the use of encrypted tunnelling through VPNs, malware detection, firewalls, intrusion detection/prevention systems and DNSSEC all considerations for an endpoint security policy.

Threats posed by attackers gaining physical access to a system requires consideration from both hardware and software security disciplines. Applications developers should employ secure coding practises, particularly when operating on any sensitive information. Care should also be taken to minimise, or if possible, to avoid the storage of secret information in physical memory. The use of software, or ideally hardware based, hard disk encryption technologies can offer protections, even when the disk is removed from a system.

Side-Channel attacks can potentially be used to reveal sensitive information. In the UniServer system, sensitive extended margin information could be targeted to create denial of service attacks or cause system instability. The variation of voltage and frequency margins, core features of the UniServer solution, may also influence the relative amount of side-channel leakages. Side-channel resilient countermeasures, employing masking and hiding strategies, should be employed to help counteract such threats.

The differing deployment architectures of full stack and bare metal are considered. In the full stack deployment, representing a micro-server data centre, the UniServer software is running under the host OS, abstracted from other guest applications under separate virtual machines. However, in the bare metal deployment, the UniServer software runs along-side other system applications. It is in this deployment architecture where the UniServer system is most exposed to interference by other applications. The UniServer log files are identified as high value assets that need to be protected from tampering, since it could potentially lead to system instability or denial of service attacks. It is therefore a recommendation that the log and policy files are stored in an encrypted format, to avoid reading and manipulation by others. Additionally, consideration should be given as to whether the files should be digitally signed, to provide assurance that they come from a trusted source. These recommendations would naturally have overheads in terms of real-time operation, so their implementation would need to be considered carefully in terms of system performance. The use of encryption, and possibly digital signing, will likely be candidates to form a security solution for the related programme deliverable of this work package, D7.6.

1. Introduction

Every networked system is a potential target for cyber-attacks. Typically, attacks are mounted remotely, exploiting one of many existing, or perhaps previously unknown zero-day vulnerabilities. In response, countermeasures and software patches are constantly being developed, to address the vulnerabilities that have been revealed. With the new edge computing paradigm, shown in Figure 1, data-centre security concepts that were originally developed with high-value, high-density cloud server installations in-mind, now need to be re-evaluated, since deployment to smaller, low cost micro-server clusters, and the many individual installations, do not warrant the high-cost physical protection schemes employed in a cloud server complex. In addition to defending against existing networking and software based attacks, further efforts need to be focussed towards physical security aspects, since edge micro-servers may often be in areas that are exposed and easily accessible, potentially enabling the direct tampering of the micro-server hardware.

This report considers system security in the context of traditional data-centre network security along with a specific focus on physical attacks, due to the increased vulnerability of direct physical access. It investigates the UniServer software architecture and the physical security risks to the sensitive extended margin values. The report is structured as follows; Section 1 introduces the context of cloud vs Edge micro-server deployments. Section 2 provides an overview of potential attacks, covering existing network/server based attacks and then physical attacks and their countermeasures. Section 3 then looks at the UniServer system software architecture, providing an analysis of the system design, along with recommendations for secure deployment. Section 4 summarises with a discussion of the main recommendations.

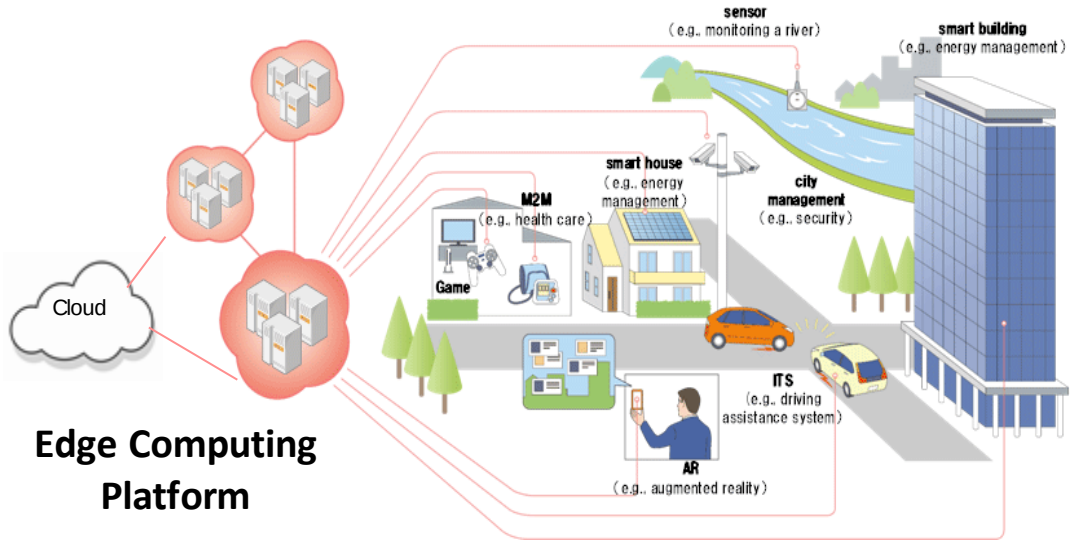

Figure 1: The UniServer edge computing deployment architecture.

1.1. Micro-Servers and Computing on the Edge

The majority of existing commercial server infrastructure is based on the cloud computing paradigm, where many high-performance servers are co-located in a high-density rackmount environment. The excess heat generated from the servers during high-throughput processing activities requires active cooling strategies, demanding significant expenditure in terms of power and cost. The global distribution of datacentres means that there may also be round-trip latency, potentially in the order 100ms – 200ms, which could be used instead to process the data locally in a lower power, more energy-efficient mode of operation.

During fabrication of server components, such as processors and memory, manufacturers will typically have a production run that has a bell-curved distribution of performance characteristics. To ensure reliable operation across the device population, and to maximise yield, operational characteristics for voltage, frequency and refresh rates are generally defined within a conservative value that is met by the lowest common performance denominators. This ensures that all devices will meet the requirement, however, it is at the expense of a loss in potential performance, or excessive power/energy consumption, compared to the regime of individual characterisation.

The UniServer project aims to develop micro-server technologies that address the above problems through development of system software that enables components to be operated within enhanced performance limits, whilst dynamically detecting, correcting and recovering from errors. As a technology that aims to reduce power on a micro-server platform, it therefore becomes an enabler for edge computing solutions, including the rapidly-increasing number of smart sensors and Internet of Things (IoT) devices which will generate a vast amount of additional sensor data that needs to be processed.

The proximity of an edge computing resource offers two main advantages; firstly, latency is greatly reduced, since data does not have to travel as far between sensor and processing node, and secondly, because of a lower deployment density of micro-servers, it offers the possibility to use less-intensive, or fully-passive, air cooling strategies, thus reducing power consumption requirements and associated costs. However, this benefit of smaller, light-weight deployment in a dispersed manner brings with it issues in terms of security, as described in [1], [2] and [3]. Outside the large data centre, as shown in Figure 2, small deployments will typically not warrant an investment in large-scale physical security such as a data centre building complex with perimeter fencing, access controls and the multitude of associated defences. In contrast, for the individual micro-server deployment, shown in Figure 3, there may be more than a light-weight housing, designed primarily for environmental protection. This opens the potential for an attacker to gain direct physical access to the micro-server and perform powerful side-channel attacks that can reveal high-value information, such as the sensitive extended margin settings of the UniServer platform.

Figure 2: Cloud Server Infrastructure in secure warehouse complex.

Figure 3: Isolated Edge Server deployment.

2. Server Security for Cloud and Edge Deployments

The UniServer platform provides opportunity for deployments at both the cloud and the edge. As networked computing platforms, both deployments will be vulnerable to a wide-range of threats and exploits that affect traditional cloud server infrastructure. In addition, edge deployments also need to be considered in the context of their exposure to direct physical access. The deployment of large numbers of individual edge servers does not offer the same economies of scale as with the co-located cloud data centre, and consequently does not warrant the associated large financial investment in physical security given to those facilities. In many cases, physical protection for an edge deployment may amount to nothing more than a light-weight cabinet aimed at protecting the system from the elements and to deter casual attempts at access or acts of vandalism. For the motivated attacker, this reduced level of security provides an opportunity to gain direct physical access and exposes a host of new threat vectors. In this section we consider the threats posed to both traditional networked server infrastructure and to the class of physical attacks, discussing the threats and countermeasures used to mitigate against them.

2.1. Server & Network Security

The primary aims of information security are to ensure the confidentiality, integrity and availability of a system [4]. There is generally no single solution to a security problem, since threats and vulnerabilities originate from many sources, rather the aim is to provide a series-layered security response, delivering defence in depth. Although this report provides analysis on specific security aspects of the UniServer architecture and system software, an overall security response should be considered in the wider sense, consisting of measures that span the range of administrative, logical/technical and physical solutions.

2.1.1. Operating System Security

The operating system (OS) is the fundamental software layer upon which the rest of the system software is built. In the common four-ring model, shown in Figure 4, the operating system is separated into two distinct regions of Kernel space, incorporating kernel memory, components and drivers from rings 0 to 2, and user space in ring 3, where end user applications may be run.

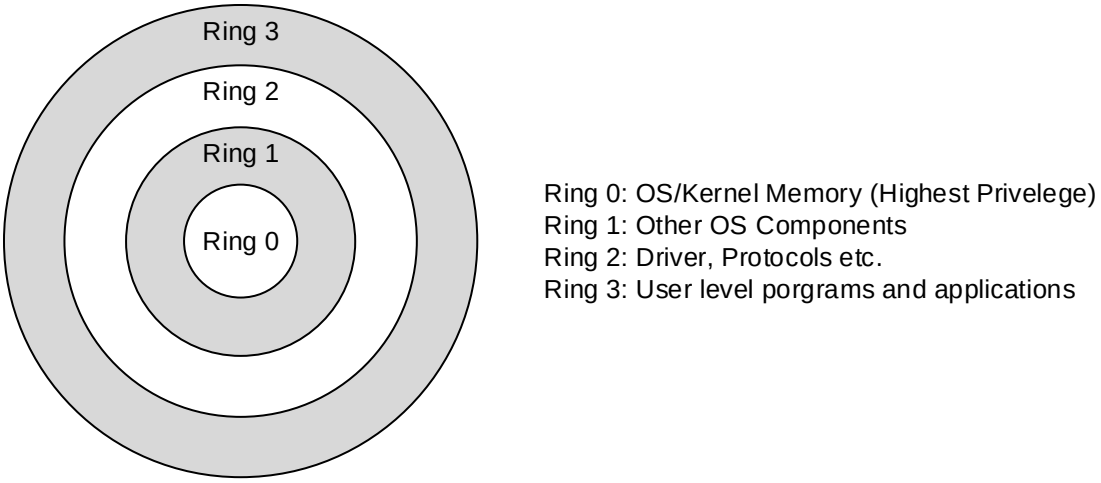

Figure 4: Layers of protection starting at Kernel Memory Ring 0, through to the User Space at Ring 3.

For most commercial operating systems, control of user access is organised under discretionary access control (DAC), providing privileges at the individual user account level. However, unlike a system under mandatory access control (MAC), where applications run in isolated memory with strong separation, typical OS's are running in a multi-tasking environment where resources are shared and are potentially accessible between applications [5]. Security is, therefore, ultimately left up to the system administrator to ensure that appropriate measures are in place and that the system is configured appropriately. Some general recommendations for operating system security, which apply to both cloud and edge deployments, are summarised below [6].

System Integrity

- Build production systems from a known and repeatable process to ensure system integrity.
- Check systems periodically against snapshots of the original system.
- Use available third-party auditing software to check system integrity.
- Backup system resources on a regular basis.

User Accounts

- Limit the number of user accounts.
- Ensure that only a few trusted users have administrative access.
- Assign the minimum required access permissions for the account that runs an application.

Password Policies

- Require the use of secure passwords, i.e. passwords of sufficient length, using a mix of letters, numbers and symbols. Don't re-use passwords and avoid the use of any personal information or dictionary words.
- Use automated tools to try and crack any weak passwords and require their update by users.
- On a UNIX operating system, activate the shadow password file.
- Use two-factor authentication.

File System

- Deny access by default.
- Provide minimal access rights where necessary e.g. read only.

Network Services

- Provide the minimum number of required services.
- Reduce the level of access permissions for network services users.
- Ensure that user accounts that have access to the Web server do not have access to shell functions.
- For UNIX/Linux, ensure that unused services do not exist in the rc files, rc0-rc6, in the /etc directory.
- Ensure that unused services are not running, and that they don't start automatically on MS Windows.
- Reduce the number of trusted ports specified in the /etc/services file.
- Protect your system against NetBIOS threats associated with ports 137, 138, and 139.
- Use wrapper services, such as iptables
- Avoid using services that have a GUI, since such services introduce many known vulnerabilities.

System Patches

- Run the latest, vendor-recommended patches for the operating system.
- Schedule regular maintenance of security patches.

Operating System Minimisation

- Remove non-essential applications to reduce possible system vulnerabilities.
- Restrict local services to those required for operation.
- Implement protection for buffer overflow.

Logging and Monitoring

- Log security-related events, including successful and failed logons, logoffs and changes to user permissions.
- Monitor system log files.
- Use a time server to correlate time for forensics.
- Secure the system log files by restricting access permissions to them.
- Secure the logging configuration file.
- Consider the use of a remote server for storage of logging information.
- Enable logging of access requests on web servers.

2.1.2. Hyperjacking

Hypervisor technology enables the deployment of numerous virtual machines (VMs) on the one system, indeed it is a key concept in shared cloud infrastructure. However, the deployment of multiple systems adds complexity and consequently the possibility for new exploits. The term virtualisation escape, or VMEscape, refers to the process by which an attacker can escape the confines of the virtual environment and is then able to exploit the host OS. Virtualised systems should therefore still be deployed under the supervision of firewalls, while guests with differing security levels, such as DMZ and internal, should not be combined on the same host.

It has been reported that malware rootkits have also been developed that act as hypervisors, installing themselves below operating systems, in a process referred to as hyperjacking. Since this software operates ostensibly outside the scope of the operating system, it can evade malware scans and also spy on the system, gathering information such as logging of passwords. In 2009, researchers from Microsoft and North Carolina State University revealed Hooksafe [7], a hypervisor class anti-rootkit, aiming to demonstrate the provision of generic protection against kernel-mode rootkits.

2.1.3. Network Attacks

Access via network ports forms the basis of most remote attacks on cloud based infrastructure. The ports of machines around the world are continually being probed to see if any ports have been left open or unsecured. It is therefore a basic preventative measure to close any unused ports and restrict access and secure those essential ports that are required to remain open. Improperly implemented TCP/IP stacks are vulnerable to various attacks such as buffer overflows, SYN flood attacks, denial of service attacks such as Smurf, ping and Fraggle and fragment attacks such as Teardrop to name but a few. These attacks can be largely mitigated by applying the appropriate configuration to disable services and apply the relevant patches.

Under the assumption that edge deployed servers are more exposed, there are numerous means by which the traditional networking security elements of firewalls, proxies, virus scanners can be circumvented, creating a means by which other nodes of the network may be exposed. In 2014, the Gameover Zeus (GOZ) botnet was responsible for the global distribution of the CryptoLocker ransomware which encrypted the victims hard drive and required payment to receive the decryption key.

Since network connections could be exposed, the communications channel of an edge device should be considered untrustworthy, since attacks such as eavesdropping on network traffic, man-in-the-middle, modification or replay attacks are all possible. It is recommended that an encrypted VPN tunnel should be used between the edge server and other elements of the network to mitigate against such attacks.

DNS hijacking exploits the vulnerability in the way local or caching DNS servers obtain information from root servers regarding the identity of the authoritative servers for a domain. It is possible for an attacker to send falsified replies, and thus control the domain resolution, forwarding the user to the attacker's server [8]. The most effective countermeasure against DNS hijacking is to upgrade DNS to Domain Name System Security Extensions (DNSSEC).

When considering the above attacks, it is evident that edge deployments edge deployments should incorporate their own endpoint security, consisting of elements such as inbound/outbound firewalls, malware scanning and intrusion detection/prevention systems as necessary security countermeasures.

2.2. Physical Attacks & Countermeasures for Edge Deployments

We now turn attention to the scenario highlighted in the introduction to this chapter, where we can assume that a determined attacker has been able to bypass the limited protections of an enclosure and has gained direct physical access to the system, providing an enhanced ability to tamper with the system. There are many such physical attacks referenced in the literature, here we aim to give an overview of attacks, giving examples for the most relevant and practical attacks, along with examples of suggested countermeasures to those attacks.

2.2.1. Memory Attacks

High-performance, processor-based, systems will generally include the following types of memory; L1/L2/L3 cache, DRAM, Flash Firmware and Hard-Disk Drives. Each of these is a potential threat vector for an attacker.

2.2.1.1 Timing Attacks

Timing attacks exploit the differences in time required to perform specific operations. For example, the time required to calculate division and multiplication instructions, or the time necessary to fetch data when a cache hit, or cache miss, is experienced. Similarly, the difference in timings when conditional branching is used, or when optimisations are used by a programmer to skip unnecessary operations, may improve application performance but at the same time can reveal sensitive information about underlying code and values being processed. A classic example was shown by Kocher in [9] where the timings for modular multiply operations in exponentiation operations, and modulo reductions of the Chinese Remainder Theorem (CRT) optimisation in RSA, could lead to the discovery of the entire encryption key on a PC.

An example of a remote network-based attack is that of Bernstein in [10], demonstrating a timing attack on OpenSSL AES, on a UNIX x86 server. The server was profiled using a known key to determine the timing characteristics for the input plaintext values. During the attack, plaintexts were sent to the server, with their timing profiles compared to the profiled reference. The information leakage was reported to be due to the non-constant timing of table lookups.

Cache-timing attacks were first proposed by Page in [11] and demonstrated by Tsunoo *et al.* in [12], where DES was broken with a >90% success rate. In [13] Tromer *et al.*, showed that the full AES key could be extracted using DM-CRYPT disk encryption on Linux with only 800 accesses to an encrypted file. The attack took 65 ms of measurement time and 3 seconds to analyse. The OpenSSL library was also attacked in as little as 13 ms, with 300 encryptions.

Countermeasures to timing attacks generally aim to perform operations in constant time. However, this is not a straight-forward task since compilers can often provide optimisations that affect timing behaviour. In addition, cache hits and variances in instruction timings are generally outside the control of the software designer. A clock-skipping countermeasure was initially proposed by Kocher in [14] which inserted random delays, to try and break up characteristic timing patterns, but this was later shown to be equivalent to adding noise to the power waveforms and could be overcome by analysis with a larger number of traces.

In [13], Tromer *et al.* considered various countermeasures against cache attacks. They suggested:

1. Avoid the use of memory accesses by replacing lookups with equivalent logical operations. This is a possibility for algorithms such as AES. However, there will be a performance trade-off.
2. Use of a bit-slicing approach.
3. Use of a cache no-fill mode, where memory is accessed from the cache during a hit and serviced from memory when there is a cache miss.
4. Dynamic table storage, where the contents of the table lookup are cycled around in memory during encryption operations to de-correlate it.

Guidance for coding standards for cryptographic implementations in software can be found in [15]. For example, in the context of timing attacks, it is recommended:

1. Do not compare secret values on a byte-by-byte basis.
2. Avoid branching predicated on secret data.
3. Avoid the use of lookup tables indexed by secret data.
4. Avoid loops that are bounded by a secret value.

The software developer can also make use of libraries, written with security in mind, such as NaCl [16] and some processors also include custom instruction sets dedicated to cryptography, such as the Intel AES-NI instructions referenced in [17] and the ARM cryptography extensions discussed in ARMv8 [18].

2.2.1.2 DRAM Attacks

Buffer Overflow is a well-known attack which can enable execution of malicious code. Strategies to counteract this attack include the use of improved input validation and bounds checking at the programmer

level, or at the system level through approaches such as the randomisation of memory layout or the structuring of buffer memory to incorporate memory spaces, sometimes termed 'canaries', that actively monitor to detect when unauthorised overflows occur.

The purposeful use of errors, exceptions and crashes can also be used to initiate memory dumping, where the entire contents of system memory are exported to enable readout of sensitive values stored in memory. It is recommended that sensitive values should not be stored in memory in the clear, rather they should be stored in encrypted form, or represented as hashed values and compared against re-computed hashes when required.

With direct physical access to a system, such as with an exposed and isolated edge server, an attacker can potentially remove DIMM memory modules from the system board. As described in [19], the use of cooling sprays, can enable a DIMM memory module to retain memory, without error, for several minutes. The memory can then be read plugged into another system out and sensitive information. This attack has been shown to make on-the-fly software-based disk encryption systems such as BitLocker, FileVault and TrueCrypt vulnerable. One countermeasure approach would be to avoid the use of pre-computed tables of information for encryption routines, which would typically be stored in DRAM, although this will have performance penalties associated with it since the values will need to be computed on-demand each time.

RowHammer is a more recent memory attack that exploits a weakness identified in commodity DRAMs, where repeated row activations can cause bits to flip in adjacent rows. A recent attack [20] used generic memory functions such as libc, memset and memcpy for attack primitives, making the attack more accessible.

2.2.1.3 Re-Flashing Attacks

Re-flash attacks target the replacement of existing system firmware with that of compromised firmware images. This can enable attackers to circumvent protections that would otherwise be in place. Due to the low-level nature of firmware access and control, such attacks can have a powerful effect on a system. Countermeasures may include incorporating password access for flashing operations.

2.2.1.4 Hard Disk Drive Attacks

Hard drives will generally host the main operating system and the application software that loads on the system, but also potentially swap page information, which may hold sensitive information temporarily stored from primary DRAM memory. Hard disks, and particularly hot-swappable server-class drives, can be removed from a system at ease, and then connected to another system by plugging in a power and data cable. The disks can then be mounted as secondary drives to be copied, interrogated, or have additional malware or software installed. All of this outside the scope of any protection from intrusion prevention systems of the original host. It is therefore advisable to consider the deployment of disk encryption technologies, such as software-based encryption, or preferably, hardware-based total disk encryption.

2.2.2. Side-Channel Attacks

We now consider a class of physical attacks termed as side-channel attacks. These attacks target the leakage of information from a system and are primarily concerned with the discovery of the secret information such as encryption keys that underpins modern cryptographic processing. The same approach can be targeted at modelled leakages of any other high-value information that is processed in a system, for example the sensitive extended margin values discussed in section 3.3.

2.2.2.1 Power Analysis Attacks

Power analysis is a powerful technique used to obtain side-channel information from a system. The power analysis attack can be categorised into two types; simple power analysis and differential power analysis.

In simple power analysis, the individual power waveform acquisitions are observed to see if information can be gleaned from them. In the attack of [9], it was observed that a single power consumption trace could reveal the entire encryption key by simply interpreting the pattern of the power trace, since modular multiply operations in exponentiation operations took varying times depending on whether the portion of the encryption key was a '1' or a '0'.

In differential power analysis (DPA), a series of power consumption measurements are recorded whilst the device is processing the target information, typically a secret encryption key, and is then compared against a

set of hypothesised power models to determine a portion of the key. The analysis is repeated for the remainder of the key portions until the complete encryption key is recovered, enabling the attacker to decrypt any data, previously encrypted with the same key. Power consumption is typically modelled by estimating the number of '1''s in a register via a Hamming weight or Hamming distance power model. Several differing methods of statistically comparing the modelled versus measured power consumptions are commonly used, such as difference of means, distance of means and Pearson's correlation coefficient [21].

Power analysis attacks are device specific and it can take from several hundred, to several million, traces to break an implementation with a DPA attack; this dependent on the signal/noise (S/N) ratio and whether any countermeasures are present. Research has been carried out on a multitude of low frequency embedded systems, where the approach has proved very successful. The attack works best when a clean voltage signal is available, preferably from the processor core of the device, where S/N is typically optimal, however attacks can also be mounted by measuring the global power supply of a device through the voltage drop across a small resistor placed between supply and ground. There are fewer published works that address attacks on a full-scale server boards, due to the additional complexities introduced by higher frequencies of operation, lack of access to processor core voltage, and the additional noise generated by numerous system hardware elements.

Countermeasures against power analysis attacks aim to break the statistical link between the power consumption and the sensitive intermediate data values. For defence against simple power analysis, countermeasures primarily focus on disturbing the power waveform to disrupt the observable pattern, and so remove the discernible information. This can be accomplished by increasing background noise signals, introducing random insertions or delays, or by removing conditional branching and employing constant time algorithms.

Protecting a device from DPA is a much more challenging task, since this attack uses advanced statistical techniques to extract information from many traces. Countermeasures can be classed into two broad categories, namely whether they aim to hide or mask the data [22]. Hiding approaches do not attempt to change the intermediate values that are processed, rather they try to change the power waveform by applying some randomisation or by making it constant. Randomising approaches were mentioned above for simple power analysis measures and could also include approaches such as shuffling or skipping of instruction clocks. To make the power consumption constant, approaches have been proposed such as the use of dual-rail pre-charge (DRP) logic styles, which uses two wires that are complementary for each signal. Other logic styles, such as Sense Amplifier Balanced Logic (SABL) was proposed by Tiri *et al.* in [23] to provide resistance against DPA. However, these approaches require custom ASIC design with careful layout considerations and have still been shown to be vulnerable to DPA attacks.

The masking countermeasure aims to change the sensitive intermediate values by applying and then removing a temporary mask operation. A simple example being an XOR with a random value. This then breaks the link between what the power model expects and what is processed inside the device. The disadvantage of masking is that it can require the application and removal of multiple masks, for example switching between Boolean and multiplicative masks. This has a processing overhead and can be complicated to design and implement.

2.2.2.2 Electro-Magnetic Attacks

Electro-magnetic (EM) attacks [24] are a variation of power analysis attacks. They differ in the method of acquisition, which uses an electric or magnetic field probe to convert EM radiation into voltage signals that are proportional to the power consumption. The probing is generally classed as being either near-field or far-field. Near-field probing is considered to be the short-range distance that is typically less than one-wavelength from the source. At this distance, the field strength is proportional to $1/r^3$ in strength, therefore placing the probe as close as possible to the source will maximise signal strength. A more invasive attack can be to remove the chip package surface and enable a fine point-tip probe to be placed very close to the exposed integrated circuit (IC), however this requires more time and generally a laboratory environment. A less invasive approach is to rest a simple loop antenna or EM probe tip against the surface of the IC, and to use active amplification to improve signal strength for appropriate quantisation scaling during acquisition.

Far-field EM attacks work at multiple wavelength distances and typically use a high frequency directional antenna to receive signals. The waveforms being captured here have escaped the confines of the near field and are propagating over free space [25]. This form of attack would likely only be possible for exposed, non-shielded enclosures.

An EM acquisition can have advantages over that of traditional power analysis attacks. Firstly, it can have a lower invasiveness. In comparison to a power analysis attack, where a resistor may need to be soldered into place, the EM probe can often be placed in close proximity, without any evidence of tampering. Secondly, there is the possibility to improve the localisation of the probe, i.e. to position it directly around the circuitry processing the sensitive information. This can help reduce the contributions of the EM fields generated from other elements of the overall power consumption. This can improve the S/N ratio, making it easier to visually identify leakages on an oscilloscope and improves the statistical analysis.

The countermeasures of hiding and masking, discussed above, also provide general protection against both EM analysis. However, for non-invasive attacks with an EM probe, physical shielding countermeasures can offer some further resistance. In [26], Yamaguchi *et al.* applied thin magnetic film to shield an integrated circuit device and reported a 6dB reduction in magnetic field signal strength.

2.2.2.3 Profiling Attacks

Profiling, or template, attacks [27] [28] use a reference device to build a characteristic power model of a device for various test inputs. The power model can then be compared against the power consumption measurements of an identical device to reveal what data has been processed internally. The template attack can potentially reveal the secret key with as little as one power trace, however, to obtain a power model with high fidelity may require the acquisition and pre-processing of many power traces, which may be a time-consuming exercise. Masking or the randomisation of execution order could be used as potential countermeasures.

2.2.2.4 Machine Learning Attacks

Machine learning is an emerging approach to side-channel attacks. Although numerous algorithms can potentially be used, the specific feature selection and data set size have the major influence on the success of the attack. Examples of approaches are supervised learning, support vector machines, random forest, neural networks and unsupervised learning. To date most research has focussed on support vector machines [29] [30] [31], random forest [32] and neural networks [33]. Countermeasures to machine learning may include higher-order masking approaches and the use of poisoned data.

2.2.3. Fault Attacks

Fault attacks aim to induce erroneous behaviour in devices by inserting transient faults that propagate through the system and reveal secret information as a consequence. The transient nature of the targeted faults means that an attack can be attempted repeatedly, and the attack developed. This approach means that no permanent damage is caused to the device and therefore it is less-likely that any evidence remains that an attack has taken place. In [34] and [35] it was shown that faults could be induced in smart card devices by varying the system supply voltage, clock speed and ambient temperatures. Since these same characteristics are altered in UniServer, it is an area of active investigation in the project, for example in terms of generation of memory and system errors.

Fault attacks in the literature have targeted both public and private key algorithms. Consider, for example, the attack on the Chinese reminder theorem (CRT) computation in RSA of [36] and the targeting of AES in [37] and [38]. The attack of [38] demonstrating that inducing two faults in the 9th rounds of AES key scheduling was enough to break the encryption system. For active attacks, the most common approach is that of fault injections, as detailed in [39].

Countermeasures to fault injections include established techniques in communications engineering, such as the use of error codes and parity checking, along with newer proposals such as concurrent error detection (CED) which suppress the operation of a circuit when error states are detected. The aim of CED is to halt the propagation of the error to the output, where the attacker can analyse whether the fault attack was successful or not. Additional proposals for countermeasures include the duplication of circuitry, or repeated computation, to provide comparators. With duplication of hardware the cost penalty is high, whilst repeated computation potentially multiplying the execution time required. Other, more efficient, schemes have been proposed, such as suggested in [40], requiring only one parity bit for each internal state of AES. The approach detects all odd errors, and in many cases the even errors, and may be a promising approach for implementation in both the hardware and software contexts.

Proposals have also been made to secure the CRT computations of RSA. In [41], the arguments of the CRT were calculated using an approach termed efficient redundancy, where values are verified before their use in the RSA algorithm. This approach, which adds little timing overhead, improves upon previous approaches requiring full redundancy.

2.2.4. USB Port Attacks

USB ports offer many useful interfacing facilities, and are a convenient means to boot a recovery system or to load firmware updates. These ports are therefore often left easily accessible for engineering or servicing use. From a security perspective, the availability of USB ports provides attackers with a direct way to infiltrate a system and gain control. Direct access via USB can often circumvent both passive and active monitoring controls. Attacks could range from the uploading of malware, collecting sensitive information, tampering with that information before it is sent onwards, or to enable buffer overflow attacks that escalate privileges on the system. The example of malware upload by USB that has had a severe impact is the Stuxnet attack of 2009 [42], which destroyed a significant amount of high-value industrial centrifuge equipment. Specially adapted USB pen drives have also been shown to be capable of destroying the power system they are connected to, for example the 'USB Killer Device' of [43]. Countermeasures for deployed edge devices should include the routine disabling of USB access, with re-enabling of access via password protected BIOS interface. The implementation of resilient power surge protections on USB bus/ports is also recommended.

2.2.5. Out-of-Order Execution Attacks

At the time of writing, two new side-channel attacks, targeting the out-of-order execution of instructions on processors, have been announced. Meltdown [44] exploits the scenario where a speculatively executed instruction, although aborted, permits the bypassing of memory protections and thus the ability to read Kernel memory from user space. The attack is deemed to affect Intel processors primarily. In the short-term, a patch based on the KAISER countermeasure of [45] has been released. This countermeasure re-maps the memory space in software. A more permanent solution will likely require architectural changes at the hardware level to control the order of permission checks for access to memory and improvements to memory segmentation [44].

The Spectre attack [46] exploits the use of speculative branch predictions to store information to cache memory that can then be targeted with side-channel techniques such as flush+reload or evict+reload cache attacks. The attack is considered more universal than Meltdown, and has already been shown to affect Intel, AMD and ARM processors [46]. Countermeasures against Spectre also appear difficult to implement. Simply disabling speculative execution would result in an unacceptable performance loss, whilst inserting temporary blocking instructions is also seen as a challenging task. Potential updates to processor microcode may be possible as a form of software patch, but likely to impact performance considerably [46].

3. UniServer System Architecture & Software Security Analysis

The UniServer solution controls the voltage, frequency and refresh rates for system processors and DRAM modules under the guidance of real-time monitoring and off-line predictive management software. These control abilities are facilitated via the exposure of relevant read/write access to hardware registers through a firmware and low-level driver layers in kernel space. With this level of enhanced access to sensitive system parameters, there are accompanying security considerations. In particular, the ability to set lower than nominal voltage levels has the potential to affect system stability, from causing mild recoverable errors, through to causing a full system crash and the subsequent loss of availability. The protection of these sensitive extended margin values, in terms of who can access and configure them, is therefore an important consideration.

In this section, the UniServer system stack and software architectures are analysed. Consideration is given to what information is exposed, how it travels up through the stack, and identify areas of potential weakness to be considered for a live deployment environment.

3.1. System Deployment Architectures

Two deployment architectures are analysed from a security perspective, namely they are the full stack, which represents the architecture for a micro-server data centre scenario, and then a bare metal architecture, applicable for standalone deployments. This covers the two extreme use cases, although intermediate deployments for smaller micro-server clusters could also be deployed, leveraging hypervisor technologies for virtualisation, without the addition of the OpenStack layer.

3.1.1. Full Stack

From a system perspective, the UniServer stack can be viewed as per the system diagram of Figure 5. At the bottom hardware layer, there are various sensors and registers, reporting parameters, such as system, processor and memory temperatures, voltage and errors. These values are presented in a set of memory mapped registers, which can be read and in some cases also written to. The registers are exposed via the Firmware Reliable Error Detection layer and the I2C bus.

At this hardware layer, access to the registers and their values is generally protected behind the firmware layer, however, as part of the UniServer development, there is an ability to directly read/write many of the sensitive registers via the system I2C bus. This I2C access is currently available from user space command line, as detailed in section 3.1 of deliverable 4.1, issuing i2cget and i2cset commands. For example, commands such as below that read the SOC VRD temperature, and change the ACPI state in PMD0;

```
$ i2cget -y 1 0x2f 0x11 w
$ i2cset -y 1 0x2f 0xE1 0x1 b
```

This direct control of registers would enable an attacker to issue commands that directly control the value of processor and memory voltages, generating errors, making the system state unstable and unreliable or possibly cause a system crash. In a live deployment, access to I2C registers via the command line should be restricted to only those with appropriate privileges.

The next level in the stack is the Firmware Reliable Error Detection layer, which exposes the registers up through low level handlers into the Hardware Exposure Interface (HEI) driver. The HEI driver incorporates the HEI Applications Programming Interface (API) which enables applications running from user space to register for event notifications, and for the Hypervisor and HealthLog Daemons to interact with the sensitive extended margins registers. This ability to register with the HEI API is a potential attack vector for rogue applications to monitor, or potentially change, the register values. It is therefore recommended that in live deployment, that registration to the HEI API is vetted, ensuring that only authorised applications can use the API to register or issue API commands.

The UniServer software of HealthLog, Predictor and StressLog operate as Daemons in the user space. As user space entities, they are managed through the kernel, and are provided with their own system memory, segregated from other running applications. Notwithstanding any memory based attacks 2.2.1, which aim to

exploit the memory space, they remain under the protection of system memory management protections and security policies. There are however issues of security related to what files they produce, for example log and configuration files, how these files are shared between the modules, and where the files are stored on the hard drive. We will address those questions in section 3.3. , when we look more specifically at the UniServer software.

Libvirt is the interface between the hypervisor and the OpenStack. It feeds in requests to spin up VMs with specific power utilisation, processing and error tolerance requirements. As a 3rd party system, security within the OpenStack system is out of scope of this analysis, other than to say that the validity of requests coming from OpenStack through libvirt should be checked for validity and bounds of reasonableness. Although the OpenStack is shown conceptually as a direct arrow link on Figure 5, it may in-fact be hosted on a remoted server, separated by an exposed network link, and should therefore be considered as vulnerable to tampering or man-in-the-middle attacks.

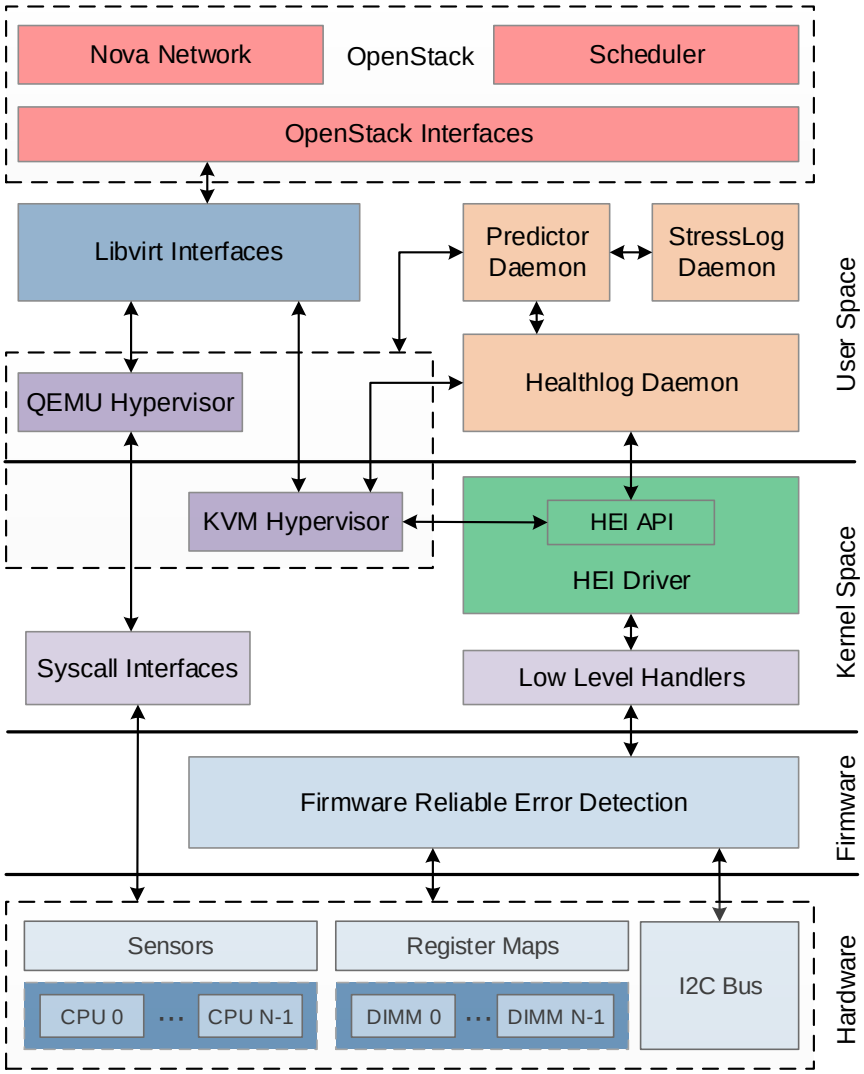

Figure 5: Full stack deployment of the UniServer architecture.

In this full stack deployment, the hypervisor offers a layer of abstraction and separation, since 3rd party software applications will be operating in their own virtual machine environment, and won't have direct access to files on the same file system as the host operating system (OS). However, as we shall see in the next section, that is not the case in the bare metal deployment. It should be noted that the recently publicised Meltdown attack of [44], discussed in section 2.2.5, has implications for virtual environments, particularly those which are not fully virtualised. For example, approaches utilising a container paradigm, where the Kernel is shared, e.g. Docker, LXC, and OpenVZ, have been shown to be vulnerable to Meltdown and permit attacks that leak data across container memory boundaries.

3.1.2. Bare Metal

The bare metal deployment, in contrast to the full stack, is configured for a standalone deployment, i.e. where there is no hypervisor or external OpenStack layers. The bare metal architecture is shown in Figure 6. In comparison with the full stack of Figure 5, it is observable that the low-level layers of Hardware and Firmware are essentially unchanged, with information flowing up through the low-level handlers into the HEI Driver. In the absence of the hypervisor, the predictor daemon now connects directly to the HEI API.

In the top layer of Figure 6, above the HealthLog daemon, are the user applications, highlighting that they are running on the same host OS and users space as the UniServer software. Although the applications will all be managed by the kernel with separate physical memory space, they will potentially be able to view and access the same directory structure on the hard drive enabling access to critical files such as log, policy or configuration files that are stored there. This could then provide an open target for malicious actors that are able to access the system. For this reason, it would be prudent to apply stringent access and privileges rights for users and applications running on the UniServer system.

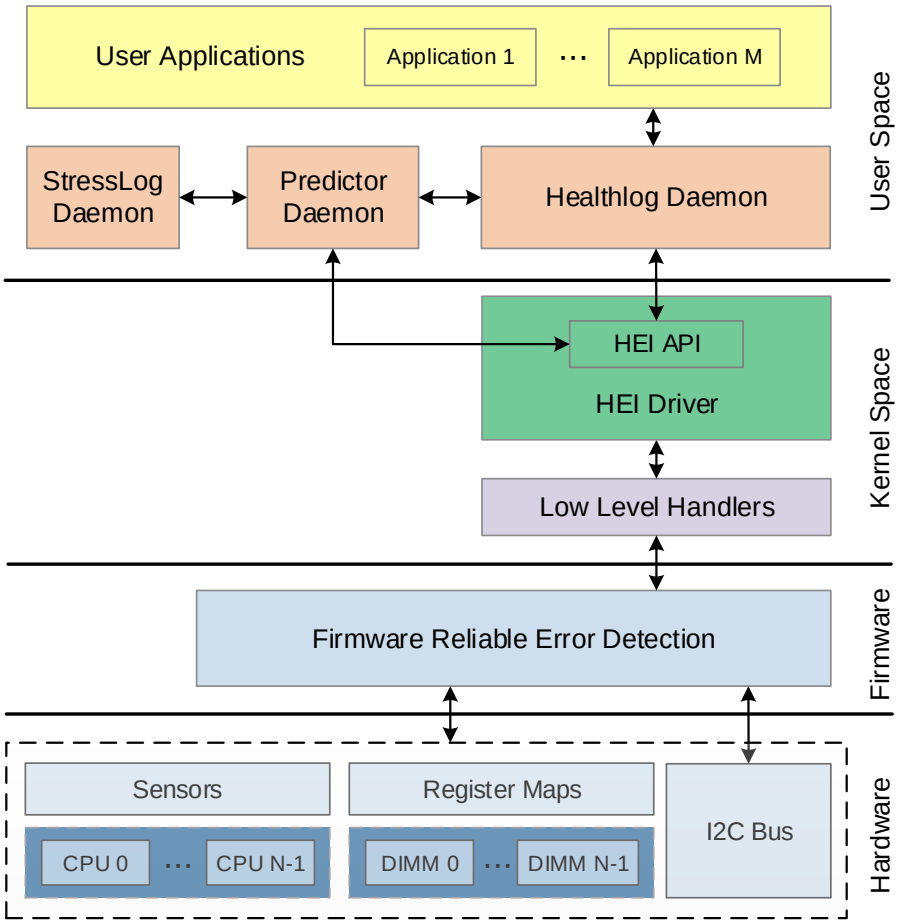

Figure 6: Bare Metal deployment of the UniServer architecture.

Although the bare metal implementation is the simplest, both conceptually and in terms of implementation, it may be considered an advantage to deploy a hypervisor to better separate the host OS and UniServer software from 3rd party applications. Of course, this is not a full solution in-itself, since an attacker with physical access to a system can access all areas and, for example, connect to system management ports such as serial and network ports, connect to hard drives ports and access the data bus directly, or mount side-channel attacks to steal sensitive information. For this reason, it is recommended that additional levels of security, such as the use of encryption and/or signing of the files that contain the sensitive extended margin settings to ensure it has not been compromised or tampered with.

3.2. System Processor and Memory Segregation

In normal full stack operation, the host operating system runs with KVM and QEMU hypervisor, with one processor module, i.e. two processors, reserved for their use. For the secure OS region, where stability is important, a lower nominal operating frequency may be set. The hypervisor can now allocate VMs to the remaining cores, setting the frequencies as requested from OpenStack. The processor segmentation is shown in Figure 7.

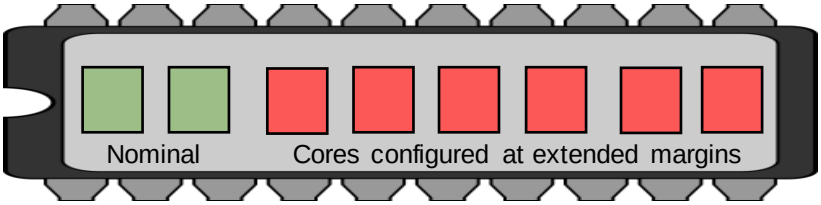

Figure 7: Processor segmentation for Host OS and extended margin cores.

Similarly, it is proposed to have a heterogeneous-reliability DRAM framework, as shown in Figure 8. This is implemented by first disabling interleaving of the memory to have read/writes, thereby in contiguous read/write block under the same voltage and refresh rate regimen. A bank of memory is then allocated for critical data usage, which includes use by the host operating system, but may also include other memory tasks from other applications that require error free performance. This first memory bank is operated at nominal voltage and refresh values to ensure reliable operation. The remaining memory bank(s) can then be operated under marginal conditions to gain power savings. This memory architecture may have potential security issues due to ability of 3rd party applications to share the host operating system memory bank. Memory attacks, such as buffer overflows, could permit modifications to sensitive regions of memory. Countermeasures for such memory attacks are discussed in section 2.2.1.

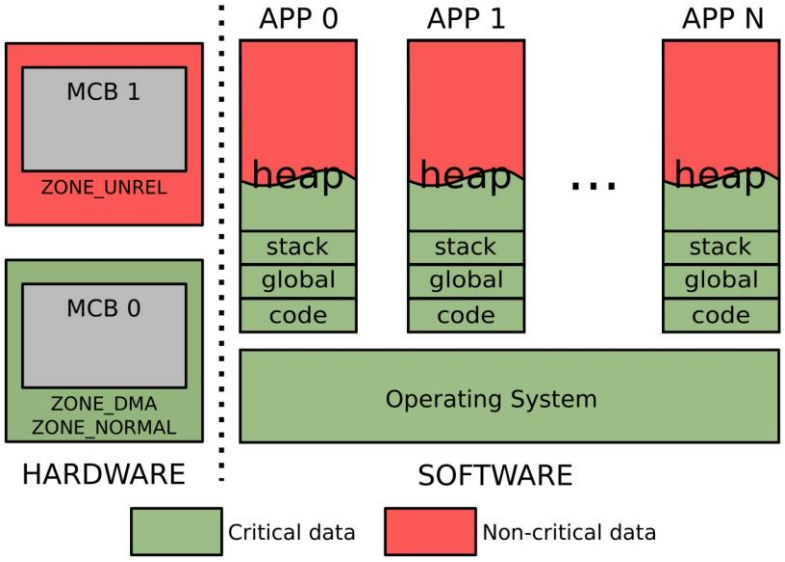

Figure 8: memory segmentation for critical and non-critical data.

3.3. System Software

In section 3.1.1, the UniServer full system stack was reviewed, with the HealthLog, StressLog and Predictor shown to operate as daemons in the user space. In Figure 9, a more detailed representation of the software components can be observed, including further details on log and policy/configuration files that are used by the software to accomplish the management activities.

The central hub of the UniServer ecosystem is the HealthLog, which acts as a gateway for messages coming up through the HEI, through the hypervisor and OpenStack and also through to the Predictor and total cost ownership (TCO) modules. The HealthLog operates in two primary servicing modes; firstly, it provides an

event-driven service, where it collects system event notifications sent up from the firmware layer, for example system error notifications, and then also provides an on-demand service in support of the predictor and StressLog modules.

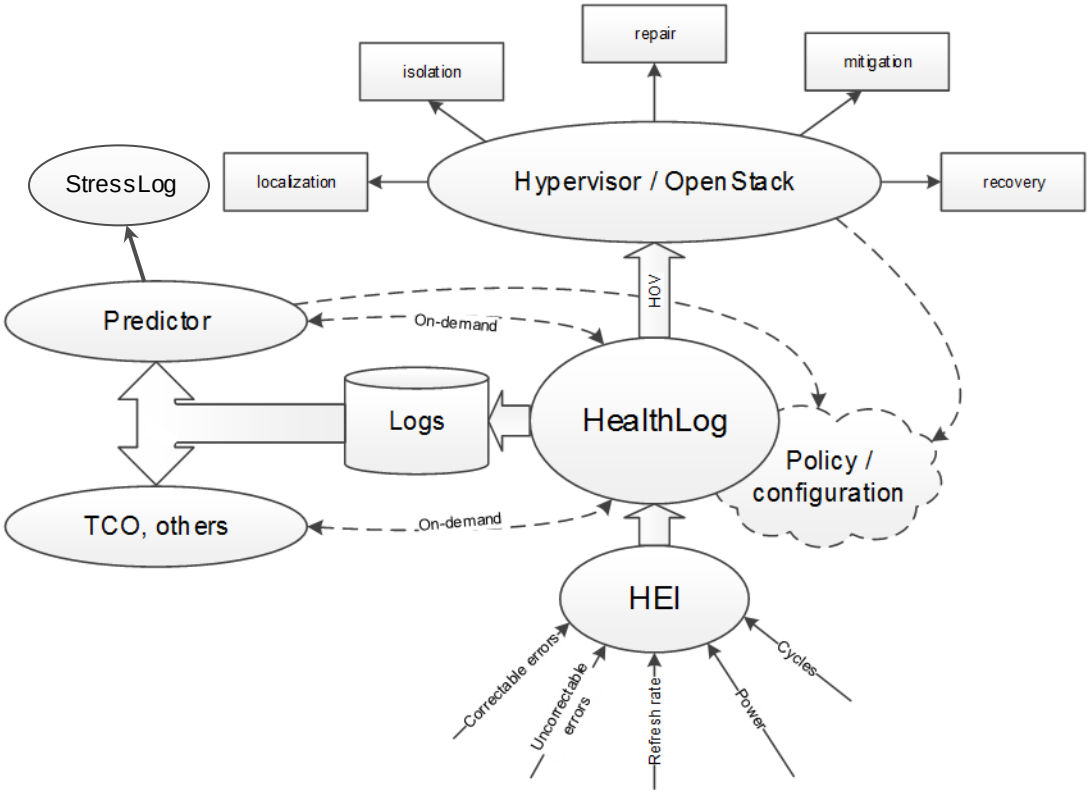

Figure 9: UniServer system software interaction model.

HealthLog monitoring gathers information categorised as reliability, power, thermal and performance metrics. In regular time intervals (or on demand), the HealthLog monitor will produce an output information vector containing the current state of these metrics, which will then be recorded within log files. These log files can then be queried by other elements, such as the Predictor and TCO. The log files will also be used by OpenStack agents on the server to guide placement of virtual machines across the cloud infrastructure.

HealthLog operates on files as shown in Figure 10. In the event handling servicing mode, notifications are streaming in from the HEI interface, via the registered HEI API interface. These values are parsed from the text stream and placed into the information vector, which is then written to the log file. At the same time, this event information is compared against configuration policy to determine if current event state warrants further action, such as notifications to other system software. The on-demand query handling is used to query specific information from the system state, or to initiate any on-demand updates of register values. The on-demand process uses system /dev/ht as the interface. The HealthLog log file is now available for use by the Predictor or TCO to analyse the history of the system state to make analyses and predictions on whether system state needs to be changed, whether StressLog needs to be run, or whether cost/performance metrics are being met.

The StressLog monitor is spawned either periodically during a machine's lifetime, for example to compensate for any aging effects on the hardware, or is triggered by higher system layers, such as the Predictor in the case of anomalous machine behaviour. In that event, the machine being tested will be taken offline and will initiate the stress test scenarios, using the provided stress target parameters. The StressLog monitor also includes a workload suite, consisting of different benchmarks and kernels that either represent real-life applications or are synthesised to stress specific components of the system. During a stress test, the HealthLog monitor executes in parallel to record system events such as errors, system values, sensors and performance counters. The StressLog monitor takes the output of HealthLog and wraps all information into a vector to be passed to the higher layers.

The Predictor is a module that utilises offline characterisation of data to predict the probability of failure for non-nominal voltage-frequency states and DRAM refresh rates. The availability constraints, provided by OpenStack, define the desired number of cores and the operating frequencies. The predictor estimates the most energy efficient voltages and DRAM refresh states whilst avoiding violation of the constraints.

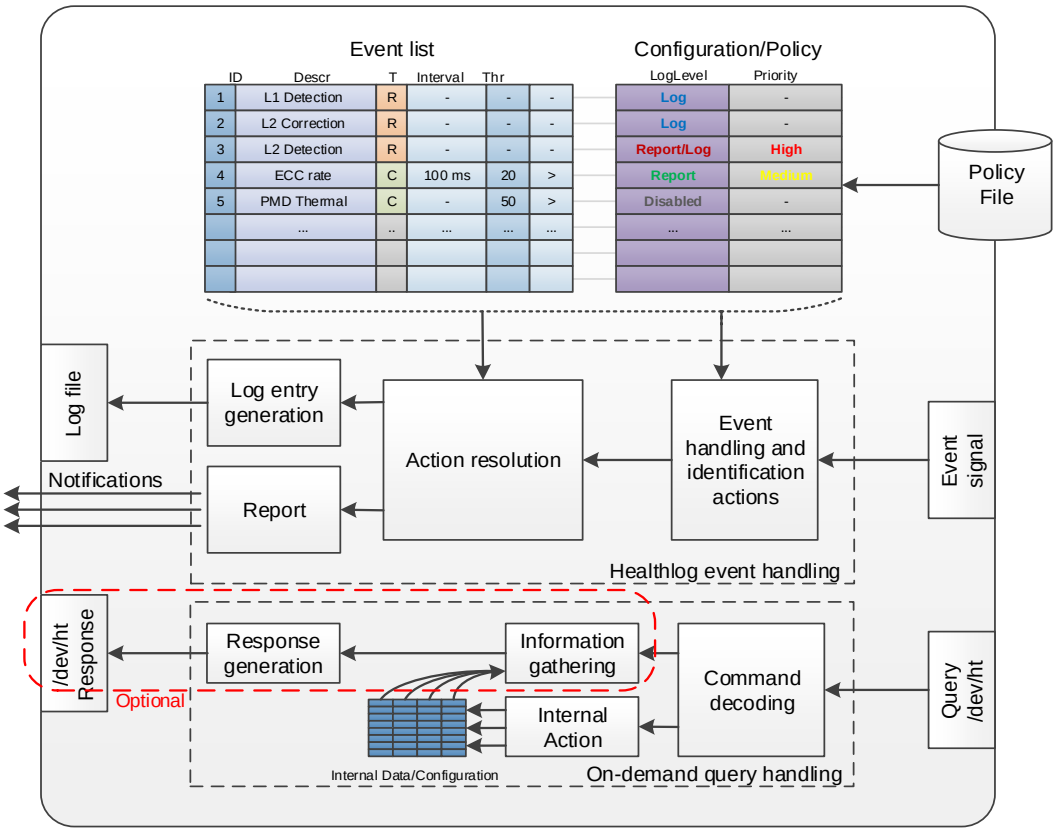

Figure 10: UniServer log file and notification event processing.

An important consideration from a security perspective is where the log and policy/configuration files sit. Although the UniServer software Daemons run in user space, and therefore in system RAM, the log and policy files are stored on the hard drive in the clear. Since the HealthLog log file is the means by which the Predictor (and therefore StressLog) gain their information, it is important for the stability of the system that the information held in this file is not corrupted or tampered with by a malicious user.

It is not difficult to imagine that if the HealthLog information values are tampered with or corrupted, that the Predictor may then determine that the system should be taken offline for StressLog re-characterisation and subsequent restoration. This process could be repeated, affecting availability and possibly full denial of service. It is a recommendation, therefore, that the log and policy files are stored in encrypted format, to avoid reading and manipulation by others. Additionally, consideration should be given as to whether the files should be digitally signed, to ensure their authenticity and origin from a trusted process. These recommendations would naturally have overheads in terms of real-time performance, so their implementation would need to be considered carefully in terms of system performance and operability.

A full summary of the security recommendations for UniServer is provided in the next section.

4. Conclusions

In this report we have seen that the move from cloud deployment model to the edge has implications for security. In contrast to a cloud data centre, housed within a large building complex with a significant level security, the edge deployment will constitute a large number of small clusters or individual installations, where high levels of physical security are not economically viable. In many situations, physical security of the micro-server may consist primarily of a light-weight enclosure, designed to protect the system from environmental factors and vandalism or casual tampering efforts. For the determined attacker, this may not prove to be an effective barrier and it should be assumed that a realistic worst-case scenario is that an attacker will be able to gain full access to the system. This then creates a larger threat surface, now incorporating physical attacks that can be used to compromise the individual micro-server, or potentially, the wider network.

Deployment at the edge still requires the implementation of traditional server and network security practises, such as those outlined in section 2.1.1 of this report. In addition, deployment at the edge should assume that networks are operating over untrustworthy links and therefore the use of encrypted tunnelling through VPNs, and the use of malware detection, firewalls, intrusion detection/prevention systems and DNSSEC should all be considered as forming the basis of an endpoint security policy.

The use of virtualisation is a core element of cloud and resource sharing technologies; however, it also opens the possibility for attacks exploiting VMEscape. Accommodating guests with differing security levels, such as DMZ and internal, on the same host, should be avoided.

Edge deployment should consider the further threats posed from an attacker gaining partial, or full, physical access to a system. This requires input not only from a hardware security standpoint, but also from software perspectives. Applications developers should employ secure coding practises, particularly when operating on any sensitive information, as highlighted in the discussions of memory attacks in section 2.2.1. Care should also be taken to minimise, or if possible, to avoid the storage of secret information in physical memory, since attacks such as buffer overflows and removal of frozen DRAM modules has been shown as effective means to extract information stored in the clear. User passwords, for example, should be stored as hashed values and passwords requested on demand for comparison or verification. The use of software, or ideally hardware based, hard disk encryption technologies can offer protections, even when the disk is removed from a system.

Side-Channel attacks can potentially be used to reveal sensitive information such as the extended margin information stored in the log and policy files. Indeed, the variation of voltage and frequency margins, core features of the UniServer solution, may also influence the relative amount of side-channel leakages. A countermeasure to this threat is the deployment of encryption using side-channel resilient countermeasures, such as masking, to break the statistical link between power measurements and hypothetical power models.

The differing deployment architectures of full stack and bare metal were discussed in section 3.1. ; In the full stack deployment, representing a micro-server data centre, the UniServer software is running under the host OS, abstracted from guest applications operating under VMs. However, in the bare metal deployment, the UniServer software runs along-side other system applications. It is in this deployment architecture where the UniServer system is most exposed to interference by other applications, which can potentially view and access each other's files or resources. The UniServer log files were identified as high value assets that need to be protected from tampering, since it could potentially lead to system instability or denial of service attacks. It is therefore a recommendation that the log and policy files are stored in an encrypted format, to avoid reading and manipulation by others. Additionally, consideration should be given as to whether the files should be digitally signed, to provide assurance that they come from a trusted source. These recommendations would naturally have overheads in terms of real-time operation, so their implementation would need to be considered carefully in terms of system performance. The use of encryption, and possibly digital signing, will likely be candidates to form a security solution for the related programme deliverable of this work package, D7.6.

5. References

[1] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *In Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on (pp. 1-8). IEEE.*, 2014.

[2] A. Alrawais, A. Alhothaily, C. Hu and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues.," in *IEEE Internet Computing, 21(2), pp.34-42.*, 2017.

[3] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury and V. Kumar, "Security and Privacy in Fog Computing: Challenges.," in *IEEE Access, 5, pp.19293-19304.*, 2017.

[4] M. C., Corporate Security Management, Elsevier Inc., 2015.

[5] S. Institute, "operating system security and secure operating systems," [Online]. Available: https://www.giac.org/paper/gsec/2776/operating-system-security-secure-operating-systems/104723. [Accessed 12 2017].

[6] IBM, "Business Intelligence Architecture and Deployment Guide - Securing the Operating System," [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSEP7J_10.2.1/com.ibm.swg.ba.cognos.crn_arch.10.2.1.doc/c_securing_the_operating_system.html. [Accessed 12 2017].

[7] Z. Wang, X. Jiang, W. Cui, W and P. Ning, "Countering kernel rootkits with lightweight hook protection.," in *In Proceedings of the 16th ACM conference on Computer and communications security (pp. 545-554).*, 2009.

[8] S. Friedl, "An Illustrated Guide to the Kaminsky DNS Vulnerability," [Online]. Available: http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html. [Accessed 12 2017].

[9] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and other Systems," in *Advances in Cryptology (CRYPTO '96). Lecture Notes in Computer Science, Vol. 1109, pp. 104-113. Springer*, 1996.

[10] D. Bernstein, "Cache-timing attacks on AES.," 2005.

[11] D. Page, "Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel," in *IACR Cryptology ePrint Archive*, 2002.

[12] Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri and H. Miyauci, "Cryptanalysis of DES implemented on computers with cache.," in *In Cryptographic Hardware and Embedded Systems-CHES 2003*, 2003.

[13] E. Tromer, D. Osvik and A. Shamir, "Efficient cache attacks on AES, and countermeasures.," in *Journal of Cryptology 23, no. 1 (2010): 37-71*, 2010.

[14] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis method and apparatus.," in *U.S. Patent 7,587,044*, 2009.

[15] Cryptocoding.net, "Cryptographic Coding Standards," in *https://cryptocoding.net/index.php/Cryptography_Coding_Standard*, 2013.

[16] "NaCl: Networking and Cryptography library," in *https://nacl.cr.yp.to/*.

[17] Intel, "Advanced Encryption Standard New Instructions (AES-NI)," in *https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/*, 2012.

[18] ARM, "ARMv8 Instruction Set Overview," in *https://www.element14.com/community/servlet/JiveServlet/previewBody/41836-102-1-229511/ARM.Reference_Manual.pdf*, 2011.

[19] J. A. Halderman, "Lest we remember: cold-boot attacks on encryption keys.," in *Communications of the ACM 52.5 (2009): 91-98*, 2009.

[20] R. Qiao and M. Seaborn, "A new approach for rowhammer attacks.," in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on. IEEE*, 2016.

[21] E. Brier, C. Clavier and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES*, 2004.

[22] S. Mangard, E. Oswald and T. Popp, Power Analysis Attacks: Revealing the Secrets of Smart Cards, Springer, 2007.

[23] K. Tiri, M. Akmal and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Solid-State Circuits Conference (ESSCIRC 2002). Proceedings of the 28th European*, 2002.

[24] J. J. Quisquater and D. Samyde, “ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards.,” in *In Smart Card Programming and Security (E-smart 2001). Lecture Notes in Computer Science, Vol. 2140, pp.200-210.*, 2001.

[25] D. Agrawal, B. Archambeault, J. R. Rao and P. Roha, “The EM Side-Channel(s),” in *In Cryptographic Hardware and Embedded Systems (CHES). Lecture Notes in Computer Science, Vol. 2523, pp 29-45.*, 2002.

[26] M. Yamaguchi, S. Kobayashi, T. Sugawa, H. Toriduka, N. Homma, A. Satoh and T. Aoki, “Development of an on-chip micro shielded-loop probe to evaluate performance of magnetic film to protect a cryptographic LSI from electromagnetic analysis,” in *In Electromagnetic Compatibility (EMC), International Symposium on, pp. 103-108*, 2010.

[27] P. Fahn and P. Pearson, “IPA: A new class of power attacks." In Cryptographic Hardware and Embedded Systems,” in *Lecture Notes in Computer Science, Vol. 1717, pp. 173-186.*, 1999.

[28] S. Chari, J. Rao and P. Rohatgi, “Template Attacks.,” in *In Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002). Lecture Notes in Computer Science, Vol. 2523, pp. 13-28.*, 2003.

[29] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede and J. Vandewalle, “Machine learning in side-channel analysis: a first study.,” in *In Journal of Cryptographic Engineering, volume 1, number 4, pages 293—302, 2011.*, 2011.

[30] A. Heuser and M. Zohner, “Intelligent Machine Homicide: Breaking Cryptographic Devices Using Support Vector Machines.,” in *Constructive Side-Channel Analysis and Secure Design -- COSADE 2012, volume 7275 of series LNCS, pages 249—264.*, 2012.

[31] L. Lerman, G. Bontempi and O. Markowitch, “Side Channel Attack: an Approach based on Machine Learning.,” in *Constructive Side-Channel Analysis and Secure Design - COSADE*, 2011.

[32] L. Lerman, S. Medeiros, G. Bontempi and O. Markowitch, “A Machine Learning Approach Against a Masked AES,” in *in Smart Card Research and Advanced Application Conference — CARDIS 2013*, 2013.

[33] R. Gilmore, N. Hanley and M. O’Neill, “Neural Network Based Attack on a Masked Implementation of AES.,” in *IEEE International Symposium on Hardware Orientated Security and Trust*, 2005.

[34] R. Anderson and M. Kuhn, “Tamper resistance—a cautionary note,” in *In Proc. of Second USENIX Workshop on Electronic Commerce, pp. 1–11*, 1996.

[35] R. Anderson and M. Kuhn, “Low cost attacks on tamper resistant devices,” in *In Proc. of 5th Security Protocols Workshop, Lecture Notes in Computer Science, Vol. 1361, pp. 125–136*, 1997.

[36] D. Boneh, R. DeMillo and R. Lipton, “On the importance of eliminating errors in cryptographic computations,” in *Journal of cryptology 14, no. 2 (2001): 101-119*, 2001.

[37] G. Piret and J.-J. Quisquater., “A differential fault attack technique against SPN structures, with application to the AES and KHAZAD,” in *In Cryptographic Hardware and Embedded Systems-CHES, pp. 77-88*, 2003.

[38] C. Kim and J.-J. Quisquater, “New differential fault analysis on AES key schedule: two faults are enough.,” in *In Smart Card Research and Advanced Applications, pp. 48-60*, 2008.

[39] H. Bar-Eli, H. Choukri, D. Naccache, M. Tunstall and C. Whelan, “The sorcerer's apprentice guide to fault attacks.,” in *Proceedings of the IEEE 94, no. 2 (2006): 370-382*, 2006.

[40] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri and V. Piuri, “Error analysis and detection procedures for a hardware implementation of the advanced encryption standard.,” in *Computers, IEEE Transactions on 52, no. 4 (2003): 492-505*, 2003.

[41] A. Shamir, “Method and apparatus for protecting public key schemes from timing and fault attacks.,” in *U.S. Patent 5,991,415*, 1999.

[42] “The Real Story of Stuxnet,” IEEE Spectrum Online, Feb 2013. [Online]. Available: https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet.

[43] “USBKiller V3,” USBKill, [Online]. Available: https://usbkill.com/. [Accessed 12 2017].

[44] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom and M. Hamburg, “Meltdown and Spectre. Bugs in modern computers leak passwords and sensitive data.,” Jan 2018. [Online]. Available: https://meltdownattack.com/meltdown.pdf.

[45] D. GRUSS, M. LIPP, M. SCHWARZ, R. FELLNER, C. Maurice and S. Mangard, “KASLR is Dead: Long Live KASLR.,” *In International Symposium on Engineering Secure Software and Systems, Springer,*, 2017.

[46] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz

and Y. Yarom, “Meltdown and Spectre. Bugs in modern computers leak passwords and sensitive data.,” Jan 2018. [Online]. Available: https://spectreattack.com/spectre.pdf.

