João Carlos Cristo Reis

Bachelor of Computer Science and Engineering

SGX-Enabled TREDIS

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Henrique João Lopes Domingos, Assistant Professor,
NOVA University of Lisbon

Examination Committee

Chair:	Name of the committee chairperson
Rapporteurs:	Name of a raporteur
	Name of another raporteur
Members:	Another member of the committee
	Yet another member of the committee

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

July, 2019

SGX-Enabled TREDIS

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Lorem ipsum.

Acknowledgements

The acknowledgements. You are free to write this section at your own will. However, usually it starts with the institutional acknowledgements (adviser, institution, grants, workmates, ...) and then comes the personal acknowledgements (friends, family, ...).

Abstract

Intel SGX hardware is a trust-computing base for applications to protect themselves from potentially-malicious OSes or hypervisors. In cloud and other outsourced computing environments, many users and applications could benefit from SGX. However, legacy applications are not prepared to work out-of-the-box on SGX.

Previous research work have already addressed library emulated OSes targeted to execute unmodified applications on SGX, but a belief has emerged that such approaches will not be interesting in terms of performance and TCB size, making in practice that application code modifications or reengineering is always an implicit and better prerequisite for adopting SGX-enabled computing environments.

(HELP) this next paragraph enters the abstract? or just conclusions after it's done?

In this thesis we intend to study existent library OSes approaches and conduct an experimental evaluation by adopting a recent solution of a OS library emulation to be ported on top of SGX, as a fully-featured library OS that can eventually be adopted to rapidly deploy unmodified applications, with overheads comparable to applications modified to use "shim" layers. Our targeted evaluation will be conducted in virtualizing the Redis Key-Value Store, redesigning and implementing it as a SGX-enabled Trusted Key-Value Store (TREDIS).

Keywords: Intel SGX, REDIS, Trusted Execution Environment, Data Protection, Privacy, Dependability ...

Resumo

O Intel SGX é uma base de computação confiável para que aplicações se protejam de SOs ou Hipervisores potencialmente maliciosos. Em *cloud* ou noutros ambientes de computação garantidos por terceiros, muitos utilizadores e aplicações podem beneficiar do SGX. Trabalhos já realizados que abordaram a emulação de bibliotecas de SOs visaram a execução de aplicações não modificadas sobre SGX, mas surgiu uma convicção de que esse tipo de abordagem não será interessante no que toca à performance ou ao tamanho da base de computação confiável fazendo com que, na prática, tanto modificações como a reengenharia do código de aplicações estejam sempre implícitos e sejam um melhor pré-requisito para adotar ambientes de computação com SGX.

(HELP) this next paragraph enters the abstract? or just conclusions after it's done?

Nesta tese pretendemos estudar as abordagens já existentes de bibliotecas de SOs e conduzir uma avaliação experimental adotando a solução recente de uma emulação de uma biblioteca de SOs para ser usada sobre o SGX, como sendo uma biblioteca completamente caracterizada que possa eventualmente ser adotada para implantar aplicações não modificadas de forma rápida, com *overheads* comparáveis a aplicações modificadas para usar camadas *"shim"*. O foco da nossa avaliação consistirá na virtualização da base de dados do tipo Chave-Valor *Redis*, redesenhando e implementando-a como uma base de dados Chave-Valor confiável com permissões SGX (TREDIS).

Palavras-chave: Intel SGX, REDIS, Ambiente de Execução Confiável, Proteção de Dados, Privacidade, Confiabilidade ...

Contents

List of Figures

List of Tables

Listings

Glossary

aliquam	tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.
computer	An electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals.
cras viverra	metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat.
donec nonummy	pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo.
integer sapien	est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus.
lorem ipsum	dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
maecenas lacinia	nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.
morbi ac	orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
morbi dolor	nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

nam lacus	libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi.
nam dui	ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo.
name arcu	libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo.
nulla malesuada	porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis.
sed lacinia	nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Acronyms

aaa	acornym aaa
aab	acornym aab
aba	acornym aba
abbrev	abbreviation of a longer text
AEU	adipiscing elit ut
AFM	aenean faucibus morbi
AMD	a magna donec
ANP	ac nunc praesent
ATG	amet tortor gravida
AVF	adipiscing vitae felis
bbb	acornym bbb
CAS	curabitur auctor semper
CDG	curabitur dictum gravida
CEA	congue eu accumsan
CIV	consectetuer id vulputate
DIA	duis eget orci
DNM	dolor nulla malesuada
DNMC	duis nibh mi congue
DRN	dignissim rutrum nam
EII	est iaculis in
ENE	et netus et
EPA	eu pulvinar at
ESQ	eleifend sagittis quis
ESV	eget sem vel
ETS	eu tellus sit

FUP	fringilla ultrices phasellus
LID	lorem ipsum dolor
LNE	libero nonummy eget
LUB	leo ultrices bibendum
LVU	lectus vestibulum urna
MAC	mollis ac nulla
MFA	malesuada fames ac
MNA	mauris nam arcu
MTS	morbi tristique senectus
NDV	nulla donec varius
NPH	neque pellentesque habitant
OER	orci eget risus
PEV	purus elit vestibulum
PIS	placerat integer sapien
PQV	pretium quis viverra
SAO	sit amet orci
SNE	sem nulla et
STC	sit amet consectetuer
TEM	turpis egestas mauris
ULC	ut leo cras
UPA	ut placerat ac
VAE	vehicula augue eu
VMR	viverra metus rhoncus
xpto	and extension of a xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto

Symbols

*

Chapter 1

Introduction

1.1 Topic Framework and Motivation

The *novathesis* was originally developed to help MSc and PhD students of the Computer Science and Engineering Department of the Faculty of Sciences and Technology of NOVA University of Lisbon (DI-FCT-NOVA) to write their thesis and dissertations Using LaTeX. These student can easily cope with LaTeX by themselves, and the only need some help in the bootstrap process to make their life easier.

However, as the template spread out among the students from other degrees at FCT-NOVA, the demand for am easier-to-use template as grown. And the template in its current shape aims at answering the expectations of those that, although they are not familiar with programming nor with markup languages, so still feel brave enough to give LaTeX a try and rejoice with the beauty of the texts typeset by this system.

1.2 Objective

It is up to you, the student, to read the FCT and/or NOVA regulations on how to format and submit your MSc or PhD dissertation.

This template is endorsed by the FCT-NOVA and even linked from its web pages, but it is not an official template. This template exists to make your life easier, but in the end of the line you are accountable for both the looks and the contents of the document you submit as your dissertation.

1.3 Expected Contributions

1.4 Document Organization

Chapter 2

Related Work

Encryption of data stored in a computer is now more important than ever. Today we live in a world where almost every program we want to run or any application we want to use goes through a large number of different machines, therefore splitting our data through all.

That is why it's needed to make sure all this information is unaccessible to potencial threats through encryption, regardless of which machine has the data.

Although encrypting the stored data is a step forward towards the security of data, this is not enough. When one machine runs a program, the data needs to be decrypted and stored into RAM to be currectly executed, thus creating a potencial vulnerability. There is where Trusted Computing Environments come in.

We'll look into more detail about existing technologies that approach this objective, and how they work or can work together as a way to grant confidentiality and integrity of data to the user.

(...) //TODO

In the summary, it's expected to have a good notion of what has been developed in those areas to tackle this security concerns, as well as which of this thesis main points have already been addressed by previous work. (????)

2.1 Trusted Computing Environments

We're currently in a period where we start to depend more and more on allowing other services to do our work for us. Technologies like cloud computing require users to trust these systems. Therefore it's needed something to grant some respectable level of security, as well as being able to grant trust to the user. That's were Trusted Computing Environments (TCE) come in handy. A TCE is a concept that came to grant integrity and

confidentiality to systems by forcing a certain machine to behave an expected way, and deny any unwanted access to the data while decrypted. This way, even if the system does not run in a trusted machine, it can be expected that it will execute as it should.

TCEs also protect the system against components that should always be trusted, like the host OS or even system admins, due to the priviledges they've got. This prevention avoids this components to abuse these priviledges, and thus always guarantee the normal execution of the system.

In the following sub-chapters we'll see how to achieve this properties, what components can be used to do it and also how each of them works.

2.1.1 TPM – Trusted Platform Modules

A Trusted Platform Module (TPM), proposed by the Trusted Computing Group (TCG), is a set of hardware and software technologies that aim to create trust in a local platform [14]. It is identified by an Endorsment Key-pair (EK) which is unique for every TPM, and also a Storage Root Key (SRK) that is used to protect other keys and data in the TPM.

As for the hardware, a TPM consists mainly on a chip, found usually in the motherboard of most machines nowadays, that provides and stores cryptographic keys that can be used to grant integrity and data confidentiality to the system, as well as persistent and volatile memory to store these keys.

As said before, the main objective of a TPM is to create the idea of a trusted platform. This will be provided by three main services: Encryption, Authenticated Boot and Attestation. The first one is used for pretty much every aspect related with security and privacy. The Authenticated boot consists in booting the OS in stages, as a way of keeping track of which code is trustable through the usage of Platform Configuration Registers (PCR), that store the trusted software hashes. As for the Attestation, we'll see in the next subsection.

2.1.2 TPM – Enabled Software Attestation

TPMs enable the use of Remote attestation, which is the capability of one system to determine if other system can be trusted to run a particular piece of software as expected or not.

This is made possible by having a trusted configuration of state as reference, provided by the PCRs defined in the boot sequence, followed by a remote system that proceeds to challenge the trusted platform (containing the TPM) with a nonce. Then the platform creates a message with the nonce received previously and the existing configuration and calculates an hash value for that message. With an Attestation Identity Key (AIK), the message is then signed and sent back to the remote system, which then proceeds to decrypt the message with the EK public part, that then compares the result with the hash of the nonce plus the configuration it had at the begining of the challenge.

If the hashes match, the remote system can then identify the TPM platform as a trusted platform.

2.1.3 HSM – Hardware Security Modules

Hardware Security Modules (HSM) are physical components whose main function is to provide and store cryptographic keys used to encrypt/decrypt data inside a system. HSMs can also perform cryptographic operations (e.g. encryption, hashing, etc.) as well as authenticate through verification of digital signatures and accelerate SSL connections [15].

This modules are used mainly in large environment systems (e.g. large distributed systems) where there are a lot of machines communicating with each other, therefore creating a more needed sense of security. The inclusion of this modules in these big systems is actually a good idea since HSMs can also help servers relieve the workload coming from cryptographic operations. However HSMs don't quite guarantee the idea of absolute security, but increment the cost of attacking the system.

Although HSMs grant some extra level of security to a system, there's some drawbacks. One of them is the cost, where to buy one HSM the price varies depending on the sophistication of the security, plus the cost of maintenance makes it even more expensive. Another disadvantage is the difficulty on updating the module. Let's say a weakness was found in a cryptographic algorithm, it's hard to update the software in an already functional HSM to eliminate that weakness [1].

2.1.4 Trusted Execution Environments

A Trusted Execution Environment (TEE) is an abstraction provided by both software and hardware that guarantees isolated execution of specific programs from other programs running on the same machine [2], but also from the host OS, hypervisor or even system administrators, preventing them from leveraging their priviledges and thus take advantage of the system. A TEE also grants secured storage of sensitive data, as well as remote attestation to make sure a given program runs as expected on a remote TEE.

For a user to communicate with his program running inside an isolated environment, a key-exchange between the TEE and the user takes place. This way it is ensured both integrity and confidentiality of data during further communications.

This TEE abstraction can be achieved either by using a virtual machine monitor or by running security critical software (from whole applications to little segments of code) under protection mechanisms provided by hardware [3].

In the next chapters we will be looking into more dept about this hardware protection providers, that are responsable to create this trusted environments in the systems nowadays.

2.1.5 Intel SGX

Intel Security Guard Extensions (SGX) are a set of instructions built in Intel CPUs, that allow programmers to create TEEs, called enclaves. Enclaves are isolation containers that create an isolated environment where sensitive code can be stored and executed inside, ensuring integrity and confidentiality to it. By doing so, it reduces the Trust Computing Base, in a way that most of the system software, apart from the enclaves itself, is considered not trusted.

Enclaves are mapped into private regions of memory, where only the CPU as access to, reducing the TCB to only the enclave and the CPU itself. Due to this restrictions, not even the most common system libraries can be accessed inside the enclave, since the OS is not trusted.

A system that encorporates the SGX under it's architecture is divided in two components, a trusted being the enclave, and an untrusted being the rest of the system. The untrusted component requests the launch of the enclave, where the CPU then manages to allocate the enclave in a private region of memory, made available only to that particular enclave. This portion of memory is kept encrypted in volatile memory, being only decrypted by the CPU if the responsible enclave requests for it [2].

Altough isolation is the main objective of Intel SGX, it still allows a way for both untrusted and trusted parties to communicate. This is made possible by the functions ECALL and OCALL. ECALLs are used for an untrusted component to call for trusted code in a secured way - the enclave copies the pointers to that specific code into a buffer, who is then made visible for the untrusted component, ensuring that the untrusted party can't know the real memory address inside the enclave. To communicate the other way arround the enclave calls for an OCALL, where the enclave is temporarily exited, executing then the unstrusted function needed. After that, the enclave is re-entered. This is mainly used by the enclave to access the network or to deal with I/O disk access.

2.1.6 RISC-V enabled Sanctum

Just like Intel SGX, the main objective of Sanctum is to offer strong isolation of software modules, altough following a different approach focused in avoiding unnecessary complexity, thus granting a simple security analysis. To make this possible, Sanctum, which tipically runs in a RISC-V processor, combines minimal and minimally invasive hardware modifications with a trusted software security monitor that is receptive to analysis and does not perform cryptographic operations using keys.

This minimality idea consists on reusing and slightly modifying existing well-understood mechanisms, while not modifiying CPU building blocks, only adding hardware to the interfaces between blocks, causing Sanctum to be adaptable to other processors in addition to RISC-V [17].

Sanctum is a pratical approach that shows that a strong software isolation is achievable with a small set of minimally invasive hardware changes, causing reasonably low

overhead. This approach provides strong security guarantees dealing with side-channel attacks, such as cache timing and passive address translation attacks.

2.1.7 ARM Trust Zone

ARM TrustZone are hardware security extentions offered by ARM application processors with the same finality as Intel SGX, create insolated environments where software execute in a trustable way.

To accomplish this, ARM processors implement two virtual processors backed by hardware access control, where the software stack can switch between two states called secure world (SW) and normal world (NW). The first one has with higher priviledges than the second one, therefore it can access NWs copies of registers, but not the other way around. SW is also responsible of protecting running processes in the CPU, while providing secured access to peripherals. Each world acts like a runtime environment and has it's own set of resources. This resources can be partioned between the two worlds or just assigned to one of them, depending on the ARM chip.

For the context switch between worlds, ARM processors implement a secured mode called Secure Monitor, where there's a special register responsable of determine if the processor runs code in SW or NW.

Most ARM processors also offer memory curtaining. This consists on the Secure Monitor allocating physical addresses of memory specifically to the SW, making this region of memory unaccessible to the rest of the system.

By default, the system boots always in SW so it can provision the runtime environment before any untrusted code start to run. It eventually transitions to NW where untrusted code can start to be executed. [7]

2.1.8 AMD SEV

AMD Secure Encrypted Virtualization (SEV) is the AMD approach to provide a TEE integrated with virtualization. It's a technology focused primarily on cloud computing environments, specifically in public infrastructure as a Service (IaaS), as its main goal is to reduce trust from higher priviledged parties (VMMs or OS), so that they can not influence the execution on the other "smaller"parties (VMs).

To achieve this, AMD grants encryption of memory through a technology called Secure Memory Encryption (SEM), or through TransparentSEM (TSEM) if the system runs a legacy OS or hypervisor with no need for any software modifications. After the data is encrypted, SEV integrates it with AMD virtualization architecture to support encrypted VMs. By doing this, every VM is now protected from his own hypervisor (VMM), unabling its access to the decrypted data. Although uncapable of accessing the VM, the VMM is still responsible of controlling each VM resources. [4]

Thus, AMD provides confidentiality of data by removing trust from the VMM, and creates an isolated environment for the VM to run, where only the VM and the processor

can be trusted. However it does not provide integrity of data, allowing replaying attacks to take place, and has a considerably large TCB, since the OS of each VM is trusted. [16]

2.1.9 Discussion

Write some comparison conclusions of why SGX and not the others.

//TODO

2.2 TEE/SGX Enabled Protection Against Untrusted OSes

A lot of applications these days depend on sensitive data to operate and so protecting these data is very much taken into account while designing the application.

One of the things we have to think about is the size of the TCB, and how to reduce it as much as possible without losing much of the operability of the system. Typically, the host OS is considered safe, trustworthy, although that is not always the case. A compromised OS can give complete access to this sensitive data in an application, regardless of how well designed it is, and that's why this is a major security problem to tackle in today's systems.

In the following section will be discussed how to achieve security for this particular problem, how to remove the OS from our TCB without losing functionability of the system, preferably without any major drawbacks in the system's performance.

2.2.1 Virtual Ghost

Virtual Ghost was presented in 2014 [8] as an approach to provide application security against untrusted OSes without requiring higher privileges.

To do so, it introduces the idea of ghost memory which is unaccessible to the OS to read or write, and provides the system with trusted services such as ghost memory management, key management, as well as encryption and signing services.

Virtual Ghost therefore suggests a different design approach. It uses compiler intrumentation to protect the system from external exploits of the OS, by sandboxing and using control-flow integrity and interposes a thin abstraction layer between the kernel and the hardware. Although this layer is similar to a hypervisor, it appears as a library of functions that the kernel code can call directly to interact with the hardware, thus avoiding the need of giving higher privileges than the OS itself.

By using ghost memory and, as said before, preventing the OS from reading and writing ghost memory pages rather than allowing to access this regions of memory, Virtual Ghost makes the system safe from a direct threat from the OS.

Overhead for non-secure data is also prevented since, with that type of data, system calls made by the application don't need encryption or hashing.

Virtual Ghost gives the application flexibility on choosing the encryption and hashing algorithm, so that the system can obtain the desired tradeoff between performance and security.

It is an approach that offers protection from kernel malware, while ensuring comparable performance to TEEs, while adding a layer of protection against external attacks on the kernel itself.

2.2.2 Flicker

Flicker [9] is an infrastructure that provides secured isolated execution of sensitive code while trusting a small TCB, which significantly improves the security and reliability of the code Flicker executes. Although Flicker does not achieve the same level of protection from physical tampering as secure coprocessors do, it provides the same strong guarantees by using modern commodity hardware.

This isolation idea is guaranteed by trusting as few as 250 additional lines of code and, as a result, Flicker circumvents legacy software and eliminates reliance on their correctness, therefore reducing the TCB. Once the TCB has been precisely defined, it's possible for the system to achieve reliability and security.

When Flicker starts, none of the software already executing can monitor or interfere with it's execution. Also, all traces of it's execution can be eliminated before non-Flicker execution resumes.

It also provides fine-grained attestation of the code executed to an external party, where the party using Flicker does not leak any information about the state of the system software.

To achieve these properties, Flicker needs hardware that support late launch and attestation, such as AMD and Intel commodity processors, and it's still capable of guarantee them even if the BIOS and OS have become malicious to the system.

2.2.3 MUSHI

MUSHI, or MUltiple level Security cloud with strong Hardware level Isolation, is a framework that provides hardware level isolation and protection to individual guest VMs executing in a cloud infrastructure. It guarantees confidentiality and integrity of a VM even during malicious attacks from both inside and outside the cloud environment.

MUSHI is mainly designed to deal with multi-level security (MLS) [13] systems, therefore MUSHI's main goal is to provide a trusted isolated environment for VM execution. To achieve this, VMs should be instantiated securely and should remain that way throughout their life cycle. With that in mind, MUSHI guarantees the following properties

- Trusted Execution - upon starting a VM, the integrity of both kernel and user image, as well as MUSHI itself, should be attested to the user, thus defining a trusted initial state

- Isolation - VMs running on the same machine should be isolated. As a result, both confidentiality and integrity will be provided to the user VM during execution. However it does not prevent side channel attacks

- User Image Confidentiality - MUSHI provides user image encryption with a key provided by the user itself

MUSHI depends on a very small TCB, including only the hardware, hardware virtualization, BIOS and System Management Mode. It can be implemented using modern commodity hardware containing SMM memory (SMRAM), necessary for the isolation between the host and VM. For remote attestation of VM images, it's done used a TPM. [11]

2.2.4 SeCage

SeCage [19] is an approach that aims to protect user-defined secrets from application potential threats and malicious OS through hardware virtualization, thus isolating sensitive code and critical secrets while denying hypervisor intervention during runtime. It's designed to deal with a strong adversary model, assuming that both application and OS can be compromised, while suporting large-scale software, which usually means a big attack surface.

SeCage devides the system in compartments, where secret compartments have all the functions with permissions to access and manipulate those secrets, and a main compartment is resposible to handle the rest of the code.

The main goals of SeCage are to assure confidentiality of user secrets (e.g., cryptographic keys) and to be a pratical approach with small overheads for large software systems. To achieve this, it ensures

- Hybrid analysis of secrets - provides a mechanism that prevents secrets from being disclosed during runtime, while not affecting their normal usage. For it, SeCage combines static and dynamic analysis. The first one is used to discover potencial functions related to the secrets, while the second one complements the first one, being more precise than the first one. With this, SeCage defines secret compartments, containing a set of secrets and their most common functions. This ensures that only the functions inside a compartment can access that set of secrets.

- Hypervisor protection - isolate each compartment in a completely isolated address space through hardware virtualization, making them unaccessible to the Hypervisor. Only if a function outside a compartment tries to access a secret without permission, it's the Hypervisor job to handle that violation.

- Separating control and data plane - since each compartment needs to communicate with each other, there will eventually be contact with the hypervisor. Therefore,

instead of frequent VM exits which cause high overheads, SeCage minimazes hypervisor intervention by only require it to define policies on whether two compartments can communicate (control plane) for as long as they conform to those policies (data plane)

SeCage was shown useful protecting large-scale system software from HeartBleed attacks, kernel memory disclosure and rootkit memory scanning, without adding any significant performance overhead to the application.

2.2.5 InkTag

InkTag [12] is a virtualization-based architecture that protects trusted applications from an untrusted OS, allowing trusted applications to securely use untrusted OS services. InkTag admits trust in the hypervisor, which is responsible to protect the application code, data, and control flow from the OS, allowing applications to execute in isolation, in a high-assurance process (HAP). Trusted applications can securely and privately share data without interference from the OS or other applications. Each secure application communicates directly with the InkTag hypervisor via hypercalls to detect OS misbehavior.

It introduces a concept called paraverification, which is a technique that simplifies the hypervisor by forcing the untrusted OS to participate in its own verification. As a result, the OS notifies the hypervisor when there is any update to be made to the state, which the hypervisor then checks for correctness. InkTag also ensures virtualization through hardware, which is used to grant isolation, as well as separate secure from insecure data. Access control policies can be specified by each trusted applications to access their secure files, with privacy and integrity of them managed by InkTag through encryption and hashing.

Other important aspect of InkTag is recoverability. InkTag is capable of ensuring crash consistency between file data and metadata. Metadata, which consists in hashed data, is fundamental to guarantee integrity of data therefore, with this consistency granted, InkTag hypervisor can protect the integrity of files even if the system crashes. If for some reason some data is inconsistent, the application must discard it.

2.2.6 Sego

Sego [18] is a hypervisor-based system that gives strong privacy and integrity guarantees to trusted applications, even when the guest OS is compromised or hostile, by removing the trust from the OS. Sego verifies OS services, like the file system, instead of replacing them.

To protect the application from the untrusted OS, Sego relies on a trusted hypervisor and assumes that hardware always executes hypervisor code correctly. It uses paraverification, where the guest OS communicates its actions and intent to the hypervisor, therefore making this verification of the untrusted OS behavior more efficient and easy.

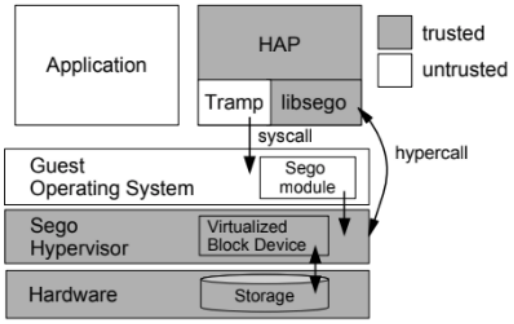

Figure 2.1: Sego Architecture Overview

Sego executes trusted code in a HAP. After booting the OS, the hypervisor starts the HAP in a way that the HAP itself can verify its own initial code and data, similar to a TPM. Once running, the hypervisor ensures that HAP's registers and trusted address space are isolated from the OS. Everytime the HAP wants to perform a system call, it must inform the hypervisor of its intent, so that the hypervisor can verify OS activity. HAPs use a small library called libsego as a way to handle system calls and get Sego services without having to change their code. Each HAP also contains an untrusted trampoline code, that uses to interact with the OS. This is used as a way to protect HAP control-flow, since it uses this trampoline as the issuer for the system calls, therefore never compromising the HAP itself.

Context switches are handled by the hypervisor, hiding any information about the HAP from the OS.

Sego does not guarantee OS availability. A compromised OS can simply shut down or refuse to schedule processes. However this is easily detected.

2.2.7 Other approaches

There are also other approaches that tackle the same problem of trusting the OS, making it impossible for the execution of an application to be compromised by a malicious OS, such as

- Hardware-assisted Data-Flow Isolation (HDFI) [**hdmiPaper**] - data isolation mechanism running on top of RISC-V that uses machine instructions and hardware to enforce isolation, by virtually extending each memory unit with an additional tag that is defined by data-flow. It grants stack protection, standard library enhancement, kernel data protection, virtual function table protection, code pointer protection and information leak prevention. It's easy to use and imposes low performance overhead, while improving security.

- Secure Channel between Rich Execution Environment and Trusted Execution Environment (SeCReT) [**secretPaper**] - it is a framework that is focused in securing

the communications between the Rich Execution Environments (REE) and the TEE built in ARM TrustZone, to add to the idea of isolation from the OS. It enables legitimate processes to use a session key in the REE, which is regarded as unsafe. To protect the key, SeCReT verifies the code's integrity and control-flow of the process every time a switch between user mode and kernel mode takes place. SeCReT's key-protection mechanism is only activated during the runtime of the process that has permission to access TrustZone, so it minimizes the performance overhead.

2.2.8 Discussion

Another conclusion about the differences of the approaches.

//TODO

2.3 SGX-Frameworks and Application Support

// TODO

Cloud providers provision thousands of computers into data centers and make them available on demand. Users rent this computing capacity to run large-scale distributed computations based on frameworks

This is a cost-effective and flexible arrangement, but it requires users to trust the cloud provider with their code and data: while data at rest can easily be protected using bulk encryption [35], at some point, cloud computers typically need access to the users' code and data in plaintext in order to process them effectively.

effectively. Of special concern is the fact that a single malicious insider with administrator privileges in the cloud provider's organization may leak or manipulate sensitive user data.

In theory, multiparty computation techniques may address these demands. For instance, data confidentiality can be achieved using fully homomorphic encryption (FHE), which enables cloud processing to be carried out on encrypted data.

However, FHE is not efficient for most com- putations

The computation can also be shared

between independent parties while guaranteeing confidential- ity for individual inputs and providing protection against corrupted parties

In some cases, one of the parties may have access to the data in the clear, while the others only have to verify the result, using zero-knowledge proofs

Still, our goals cannot currently be achieved for distributed general-purpose computations using these techniques without losing performance ——————————

2.3.1 VC3 protection for MapReduce Jobs

Verifiable Confidential Cloud Computing (VC3) [6] is a framework that achieves confidentiality and integrity of data, as well as verifiability of execution of code with good performance through MapReduce [5] techniques. It uses Intel SGX processors as a building block and runs on unmodified Hadoop [10]. In VC3 users implement MapReduce jobs, compile and encrypt them, thus obtaining a private enclave code E-. They then join it with a small portion of public code E+, that implements the protocols for key exchange and job execution. Users then upload the resulting binary code to the cloud, where enclaves containing both E- and E+ are initialized by an untrusted framework F.

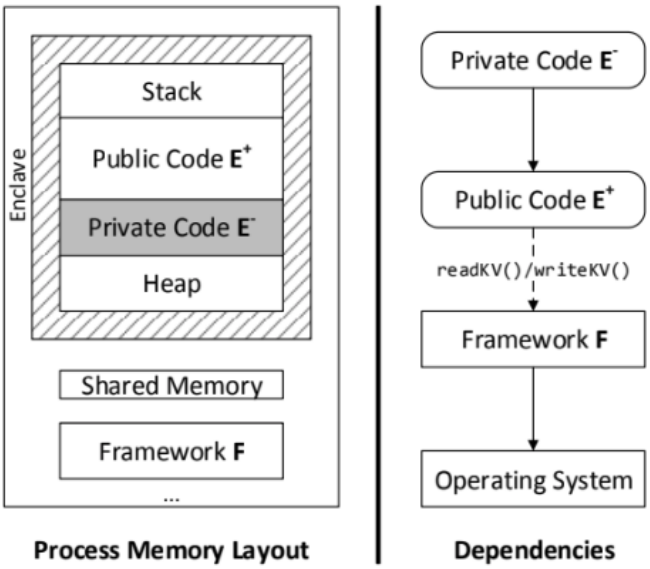

Figure 2.2: VC3 memory design model on the left, component dependencies on the right

A MapReduce begins with a key exchange between the user and the E+ code running in the enclave. After this, E+ can proceed to decrypt E- and process the encrypted data. VC3 isolates this processing from the OS by keeping an interface between the E+ layer and the outside of the enclave. This interface consists of basically two functions: readKV() and writeKV(), for reading a key-value pair on Hadoop or write it, respectively. Also, the data inside of the enclave is passed to the outside, more specifically from E+ to the untrusted F, by using a virtual address space shared by both. With VC3, both E- and the user data are always encrypted while in the cloud, except when processed by the trusted processor, while allowing Hadoop to manage the execution of VC3 jobs. Map and reduce nodes are seen as regular worker node to Hadoop, therefore Hadoop can provide its normal scheduling and fault-tolerance mechanisms, as well as load balancing. VC3 accomplishes this keeping Hadoop, the OS and the hypervisor out of the TCB.

2.3.2 Protected Zookeeper

2.3.3 Ryoan Sandboxing

2.3.4 Opaque

2.3.5 Graphene-SGX

2.3.6 Network services protection approaches

2.3.7 Application-level protection approaches

2.3.8 Discussion

The *novathesis* class can be customized with the options listed below.

---

docdegree=OPT phd(*), phdplan, phdprop, msc, mscplan, bsc

The type of the document: PhD Thesis (default), PhD Plan, PhD Proposal, MSc Disseration, MSc Plan, BSc Report

---

school=OPT nova/fct(*), nova/fcsh, nova/ims, ul/ist, ul/fc

The name of the school. This option changes the typesetting of the cover and some School specific formating, like margins, fonts, paragraph spacing and indentation, etc...

---

lang=OPT en(*), pt

The main language for the document. Currently only Portuguese and English are supported. Other languages are expected to be support in forthcoming versions.

---

fontstyle=OPT bookman, charter, fourier, kpfonts(*), mathpazo1, mathpazo2, newcent

The font set to be used in the document. Please note that a font set include definitions for the main text, headings, maths, etc.

---

chapstyle=OPT bianchi, bluebox, brotherton, dash, default, elegant(*), ell, ger, hansen, ist, jenor, lyhne, madsen, pedersen, veelo, vz14, vz34, vz43

The chapter style, i.e., the look of the chapter beginning.

---

converlang=OPT en, pt(*)

The language to be used when typesetting the cover page.

---

otherlistsat=OPT front(*), back

Where to put the other lists besides the table of contents. The default is (`front`) before the main text. But some scientific areas prefer them at the end of the document (`back`), just before the Appendixes.

---

aftercover=OPT true, false(*)

Include or don't include the contents of the "`aftercover`" file. The default is for this file to be ignored (if if it exists).

---

linkscolor=OPT darkblue(*), black

The color for all the hyperlinks in the PDF file. The "`media=paper`" option (see below) will override this option to "`black`"

---

spine=OPT true, false(*)

Generate the book spine and the last page in the PDF.

---

biblatex=OPT OPT={list of options for `biblatex`}

Customize `biblatex`, the bibliography management system used in this class. Probably you will want to change the value of the `biblatex` "`style`" option. For other customizations of `biblatex` check its manual.

---

memoir=OPT OPT={list of options for `memoir`}

*Customize the base class `memoir`. The `memoir` manual should be the first document to be consulted when looking for "**how can I do this?**" You may wnat to change the base font size from 11pt to a smaller (10pt) or larger (12pt) size. Also, remember to change the "`draft`" to final when your document is finished.*

---

media=OPT screen(*), paper

Behavior to be customized in the school options/configuration. Expected definitions for screen are: left and right margins are equal and use colored links. Expected definitions for paper are: left and right margins are different and use black links.

2.4 Related work analysis and rational

In this section we will provide some additional considerations about some of the customizations available as class options.

Chapter 3

Elaboration Work Directions

This Chapter aims at exemplifying how to do common stuff with LaTeX. We also show some stuff which is not that common! ;)

Please, use these examples as a starting point, but you should always consider using the *Big Oracle* (aka, Google, your best friend) to search for additional information or alternative ways for achieving similar results.

3.1 Materialization of objectives and contributions

3.2 System Model and Reference Architecture

3.3 Prototyping effort

3.4 Prototype Validation and Experimental Assessment

3.5 Open issues

Chapter 4

Elaboration Plan

This is the Chapter Four ERROR: File 'chapter5' does not exist!!!

Bibliography

[1] J. Attridge. *An Overview of Hardware Security Modules*. 2002. URL: https://www.sans.org/reading-room/whitepapers/vpns/overview-hardware-security-modules-757.

[2] G. R. Borges. "Practical Isolated Searchable Encryption in a Trusted Computing Environment." Master's thesis. FCT-UNL, 2018.

[3] B. M. C. Braga. "P-Cop: Securing PaaS Against Cloud Administration Threats." Master's thesis. Instituto Superior Técnico, Universidade de Lisboa, 2016.

[4] T. W. David Kaplan Jeremy Powell. *AMD MEMORY ENCRYPTION*. 2016. URL: http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf.

[5] J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. Accessed: 03-07-2019. 2004. URL: https://static.googleusercontent.com/media/research.google.com/pt-BR//archive/mapreduce-osdi04.pdf.

[6] C. F.C.G.M.P.G.M.-R.M. R. Felix Schuster Manuel Costa. *VC3: Trustworthy Data Analytics in the Cloud using SGX*. Accessed: 03-07-2019.

[7] A. W.R.A.J.C.P.E.C.F.K.K.J.L.D.M.M.N.D.R.R.S.S. T. Himanshu Raj ContainerX; Stefan Saroiu and D. Wooten. *fTPM: A Software-Only Implementation of a TPM Chip*. 2016. URL: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_raj.pdf.

[8] V. A. John Criswell Nathan Dautenhahn. *Virtual Ghost: Protecting Applications from Hostile Operating Systems*. 2014. URL: http://sva.cs.illinois.edu/pubs/VirtualGhost-ASPLOS-2014.pdf.

[9] A. P.M.K.R.H. I. Jonathan M. McCune Bryan Parno. *Flicker: An Execution Infrastructure for TCB Minimization*. 2008. URL: https://www.cs.unc.edu/~reiter/papers/2008/EuroSys.pdf.

[10] S. R.R. C. Konstantin Shvachko Hairong Kuang. *The Hadoop Distributed File System*. Accessed: 03-07-2019. URL: https://hadoop.apache.org/.

[11] W. L.Y.T. H. Ning Zhang Ming Li. *MUSHI: Toward Multiple Level Security Cloud with Strong Hardware Level Isolation*. 2013.

[12] A. M.D.M.Z.L.E. W. Owen S. Hofmann Sangman Kim. *InkTag: Secure Applications on an Untrusted Operating System*. Accessed: 28-06-2019. 2013.

[13] *Red Hat Documentation - Multi-Level Security (MLS)*. Accessed: 26-06-2019. URL: https://web.mit.edu/rhel-doc/5/RHEL-5-manual/Deployment_Guide-en-US/sec-mls-ov.html.

[14] A. M. B. Ribeiro. "Management of Trusted and Privacy Enhanced Cloud Computing Environments." Master's thesis. FCT-UNL, 2019.

[15] N. Saboonchi. "Hardware Security Module Performance Optimization by Using a "Key Pool"." Master's thesis. KTH Royal Institute of Technology in Stockholm, 2014.

[16] S. L.W. S. Saeid Mofrad Fengwei Zhang. *A Comparison Study of Intel SGX and AMD Memory Encryption Technology*. 2018. URL: http://www.cs.wayne.edu/fengwei/paper/sgxsev-hasp18.pdf.

[17] I. L. Victor Costan and S. Devadas. *Sanctum: Minimal Hardware Extensions for Strong Software Isolation*. 2016. URL: https://eprint.iacr.org/2015/564.pdf.

[18] M. Z.L.O.S.H.Y.X.E. W. Youngjin Kwon Alan M. Dunn. *Sego: Pervasive Trusted Metadata for Efficiently Verified Untrusted System Services*. Accessed: 02-07-2019. 2016.

[19] K. C.H.C.Y. X. Yutao Liu Tianyu Zhou. *Thwarting Memory Disclosure with Efficient Hypervisor-enforced Intra-domain Isolation*. Accessed: 27-06-2019. 2015.

Appendix

Appendix 1 Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus.

Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Appendix B

Appendix 2 Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus.

Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

ANNEX I

Annex 1 Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus.

Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.