#CYSARM

Bootstrapping Trust in a "Trusted" Virtualized Platform

Hagen Lauer,
Amin Sakzad,
Carsten Rudolph,
Surya Nepal

MONASH University

Introduction

Working with untrusted computers is often not possible

Trust is a known factor for adoption and proliferation

Computers need reliable roots of trust

Trust establishment is enabled by TPM as root of trust

The TPM can be used to establish trust in a system.

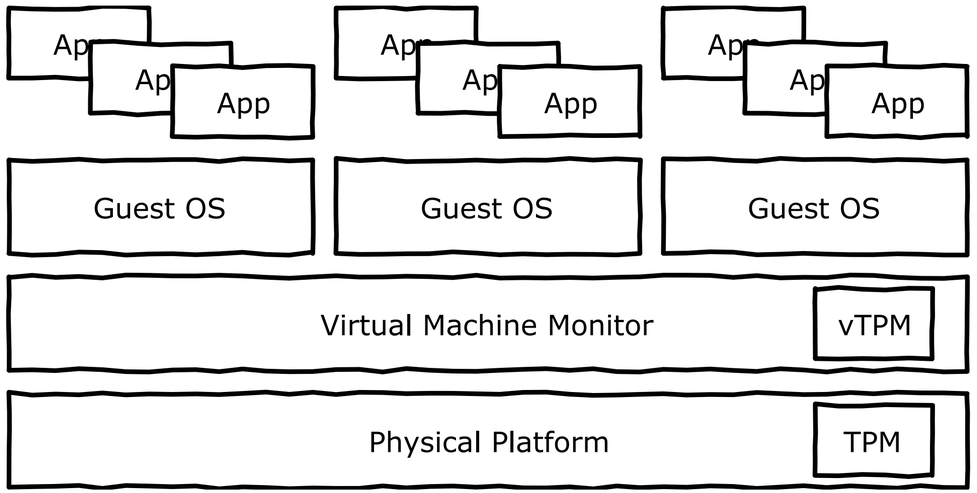
Introduction

Verifier is interested in virtual environment

Security properties of virtual environment directly depend on virtualization system

Verifier needs to establish trust in virtual environment

Verifier needs to establish trust in virtualization system

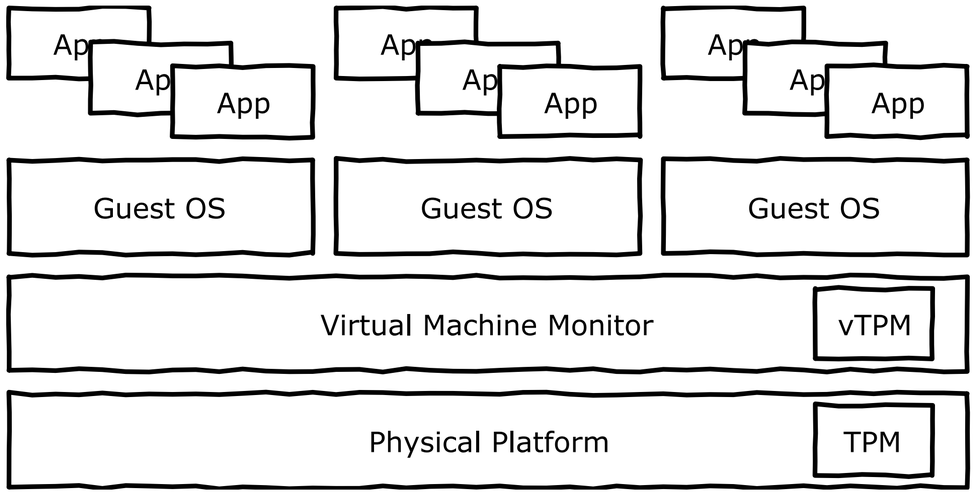


Introduction

TPM doesn’t scale with virtual environment

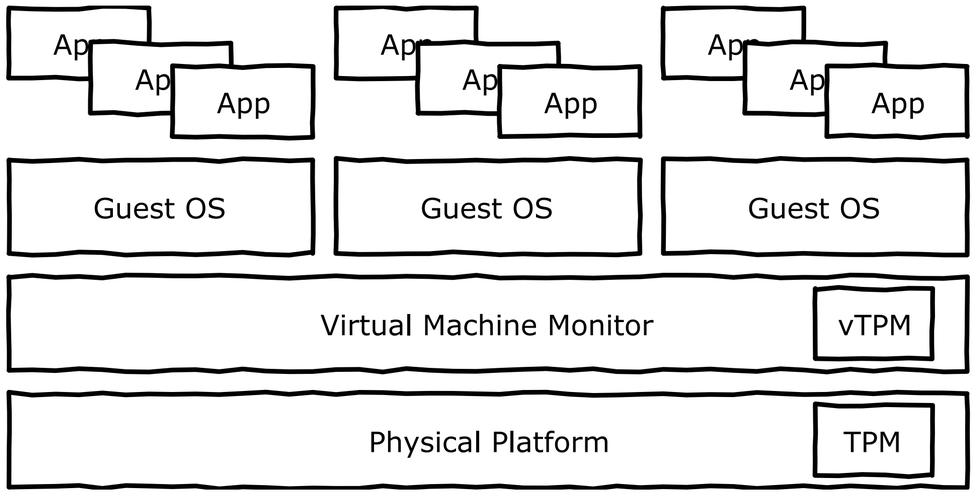
TPM is used to establish trust in virtualization system

Use vTPM as a root of trust for virtual environment



Introduction

TPM can be used to establish trust in VMM.



Can vTPM be used to establish trust in Virtual Environment?

MONASH University

Agenda

Introduction

Background

Goldeneye Attack

Discussion & Conclusions

Background and Model

Background and Model

Virtualization System (VS) enables execution of multiple computing systems on the same hardware

... is a collection of components required to reach the desired virtualization target

... supports the virtual platform

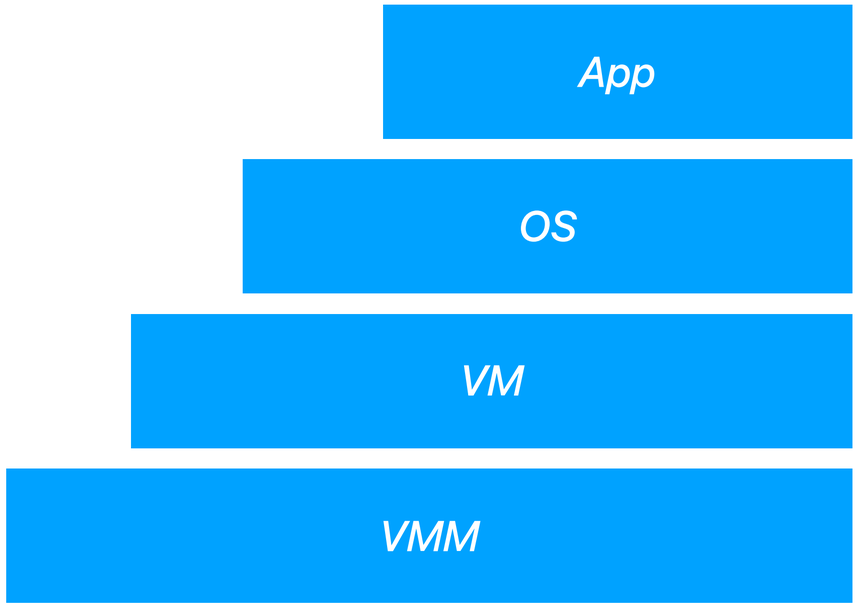
Background and Model

Process (App)

Operating System (OS)

Virtual Machine (VM)

Virtual Machine Manager (VMM)



Background and Model

TPM implements several system roots of trust

TPM effectively allows trust establishment in system state

vTPM serves as root of trust for a virtual machine

vTPM is essentially a program and a VS component

Background and Model

In summary,

Hardware + TPM forms a “trusted” platform

Virtualized Platform + vTPM forms a virtualized “trusted” platform

+

+

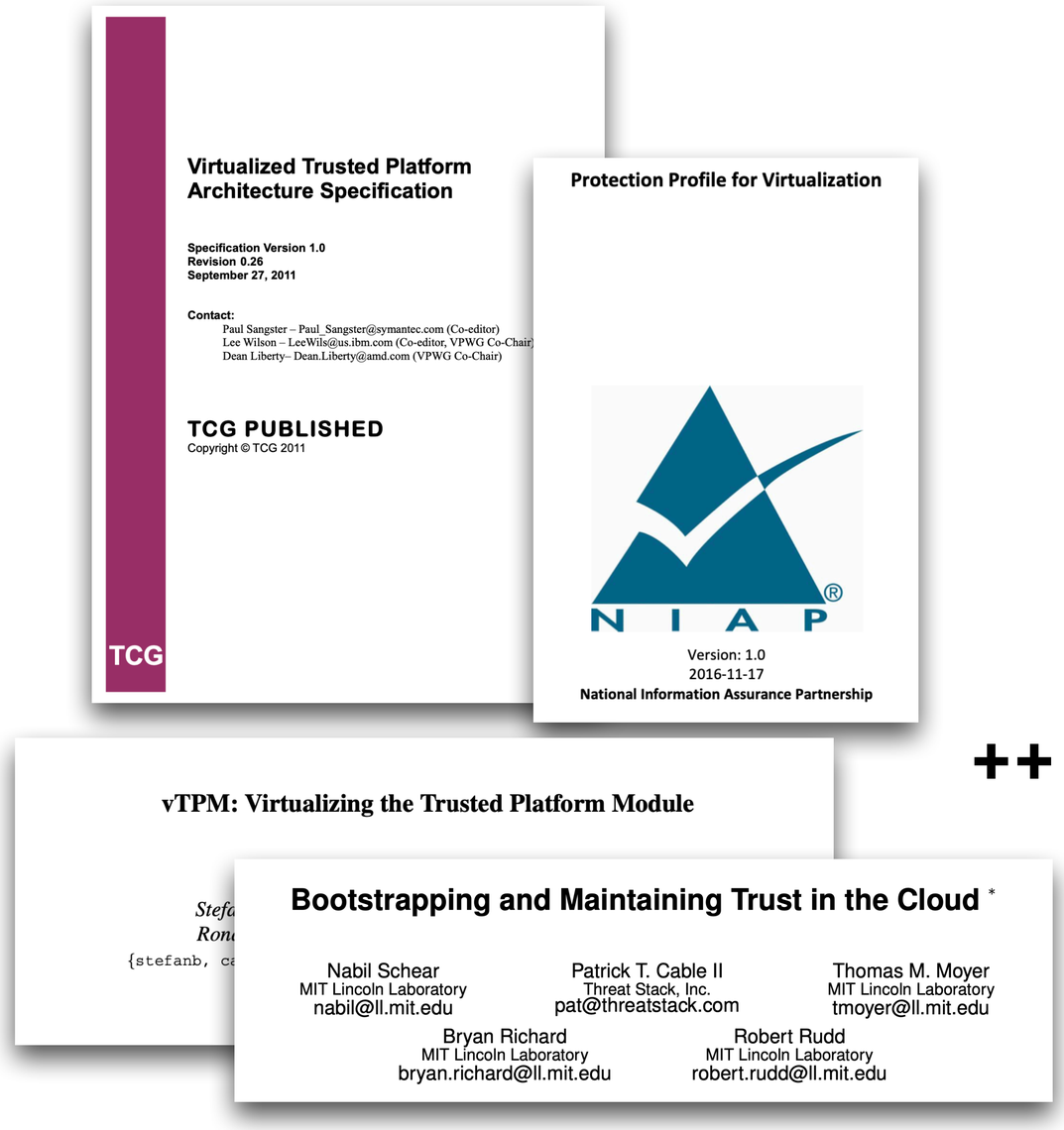
Goldeneye Attack

Goldeneye Attack

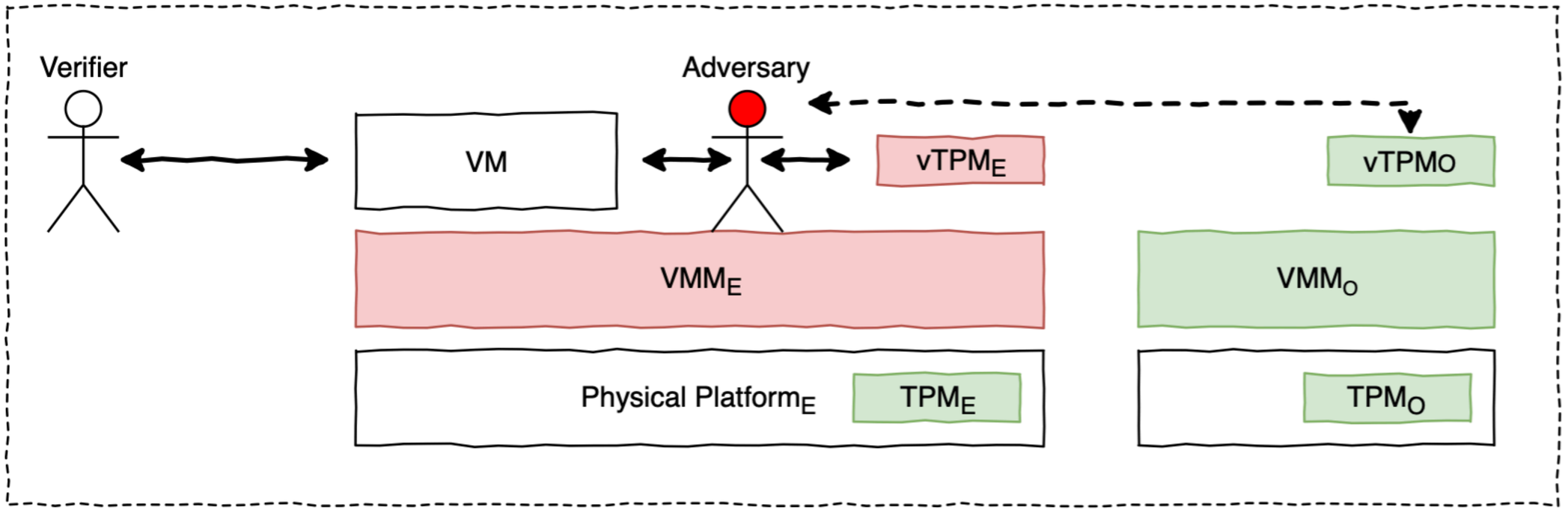
Stated goals of Virtualized Trusted Platform are:

- Enable a verifier to obtain (...) information about the security state of the virtualized environment

- Enable a verifier to obtain (...) information about the security state of the underlying virtualization system providing isolation and resources to the virtualized environment



Goldeneye Attack



a) Example attack using a weak vTPM - VM association.

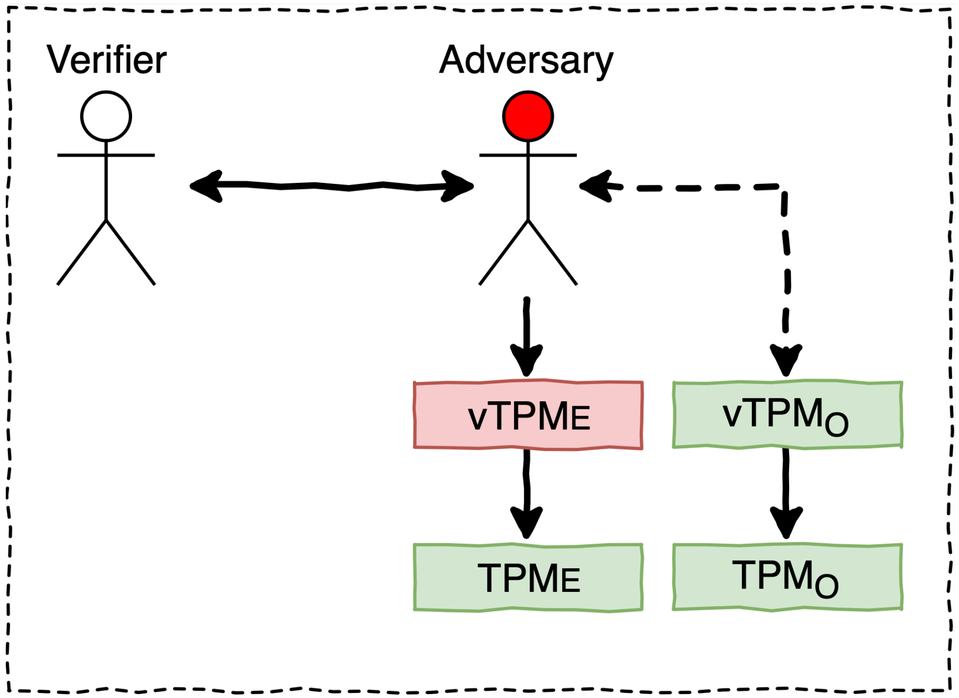
Goldeneye Attack

Goldeneye adversary has limited control over association

vTPM carries VM trust information

TPM carries VS trust information

Verifier establishes trust in the wrong but good system



b) Logical equivalent of a weak association.

Goldeneye Attack

Changing virtual Root of Trust

Adversary clones or fabricates vTPMs

Place or copy system states in vTPM

→ Allows trust establishment in a untrustworthy VM

Changing physical Root of Trust

Similar to Cuckoo attack [18]

vTPM has reference to TPM

Adversary associates a VM with vTPM on another Virtualization System

→ Allows trust establishment in another VS

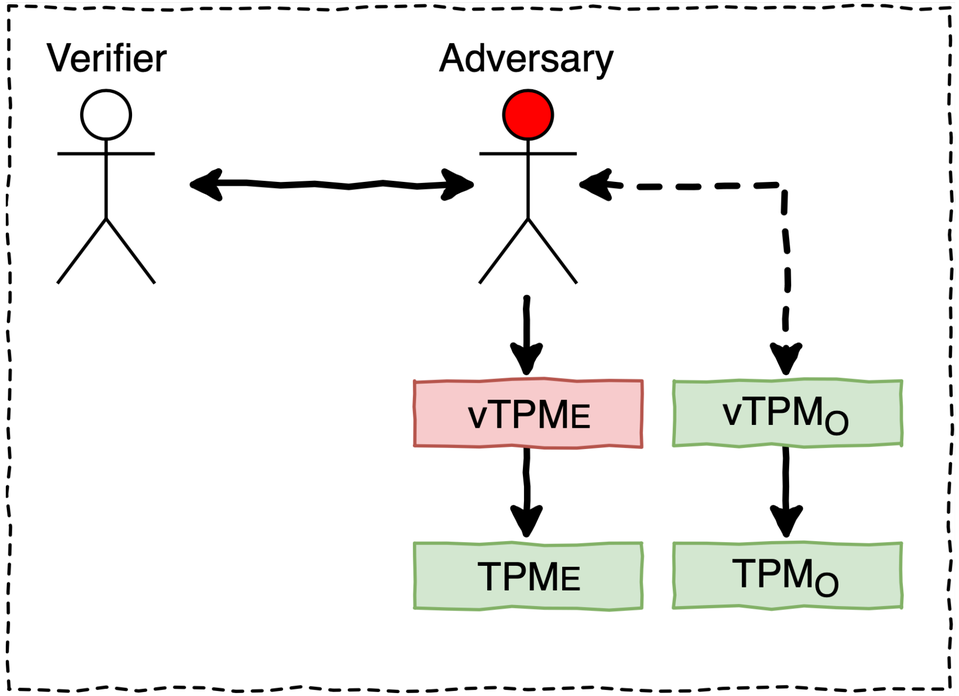
Goldeneye Attack

Exploits architectural gap in the design of virtual trusted platform

Verifier queries vTPM about VM state and eventually VS state

Relies on “strong association” between vTPM and VM [3,7,8,23]

Association itself is not verifiable nor enforceable by trusted elements



b) Logical equivalent of a weak association.

Goldeneye Attack

Informal model of virtualized trusted platform 

Outline Goldeneye attack and effects 

Describe path towards executing the attack 

Goldeneye Attack

Formal system model

Formalization of flawed trust establishment

Show that attack works and trust establishment does not

Goldeneye Attack

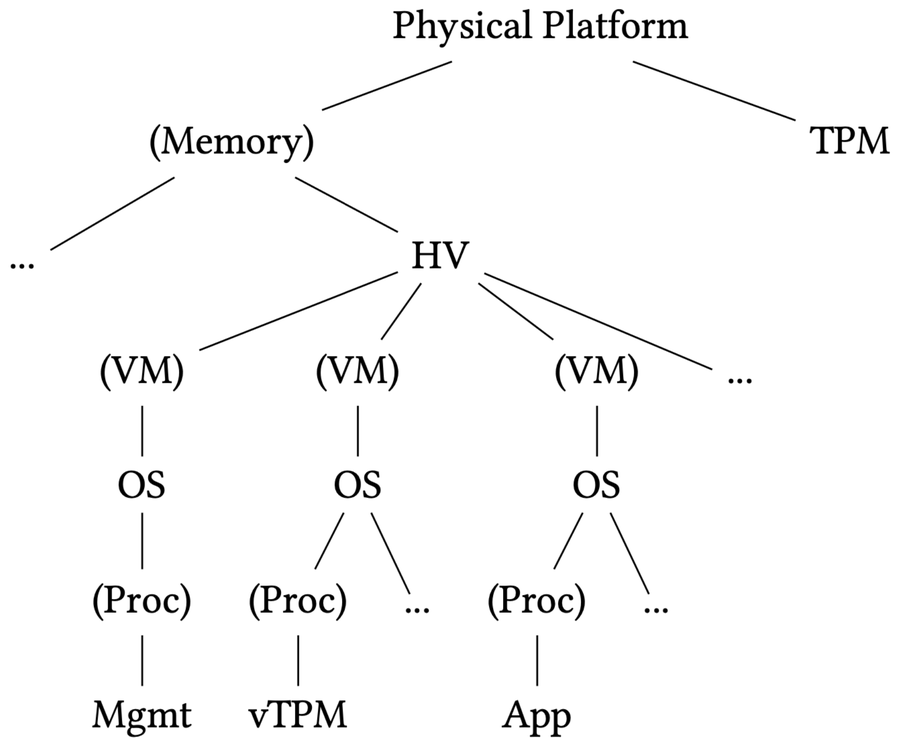
Graphical view of Virtualization System

Describe space and motion independently

Today we discuss *placing* and *linking*

Bigraph [17] as foundation and notation

Reasoning strategy for dependencies in the system graph



Goldeneye Attack

Proof strategy for Goldeneye

- Axioms for Trusted Computing System
- Axioms for Trust Establishment
- Assume an untrustworthy system
- Show that axioms and assumption lead to trusting an untrustworthy system

Predicate	Meaning
$\text{Trusted}_{\text{A}}(\text{a})$	Agent a is trusted
$\text{SaysSecure}(a, e)$	a says element e is secure
$\text{Trusted}_{\text{VS}}(vs)$	vs is trusted
$\text{On}(tpm, m)$	TPM tpm is on m
$\text{Trusted}_{\text{TPM}}(t)$	TPM t is trusted
$\text{Trusted}_{\text{VTPM}}(\text{vtpm})$	vTPM $vTPM$ is trusted
$\text{Bound}(vtpm, vm)$	$vTPM$ is bound to vm
$\text{SaysBound}(vm, vtpm)$	vm says $vtpm$ is bound
$\text{Trusted}_{\text{VM}}(\text{vm})$	vm is trusted

$$\forall a, e\ \text{Trusted}_{\text{A}}(a) \wedge \text{SaysSecure}(a, e) \supset \text{Secure}(e)$$
$$\forall t, m\ \text{On}(t, m) \wedge \text{Secure}(m) \supset \text{Trusted}_{\text{TPM}}(m.t)$$

$$\forall vm, vtpm\ \text{Secure}(vm) \wedge \text{SaysBound}(vm, vtpm) \supset \text{Bound}(vm, vtpm)$$
$$\forall vtpm, vm\ \text{Bound}(vtpm, vm) \wedge \text{Trusted}_{\text{VTPM}}(m.vs.vtpm) \supset \text{Trusted}_{\text{VM}}(m.vs.vm)$$
$$\forall t, vs\ \text{Trusted}_{\text{TPM}}(m.t) \supset \text{Trusted}_{\text{VS}}(m.vs)$$
$$\forall t, vs, vtpm.\ \text{Trusted}_{\text{TPM}}(m.t) \wedge (m.vs.vtpm) \supset \mathit{Trusted}_{\mathit{VTPM}}(m.vs.vtpm)$$

Goldeneye Attack

Generic model for virtualization systems

Predicate logic for trust establishment

Link system graph with predicate system

Proof of two related attacks on weak vTPM-VM association

Discussion & Conclusions

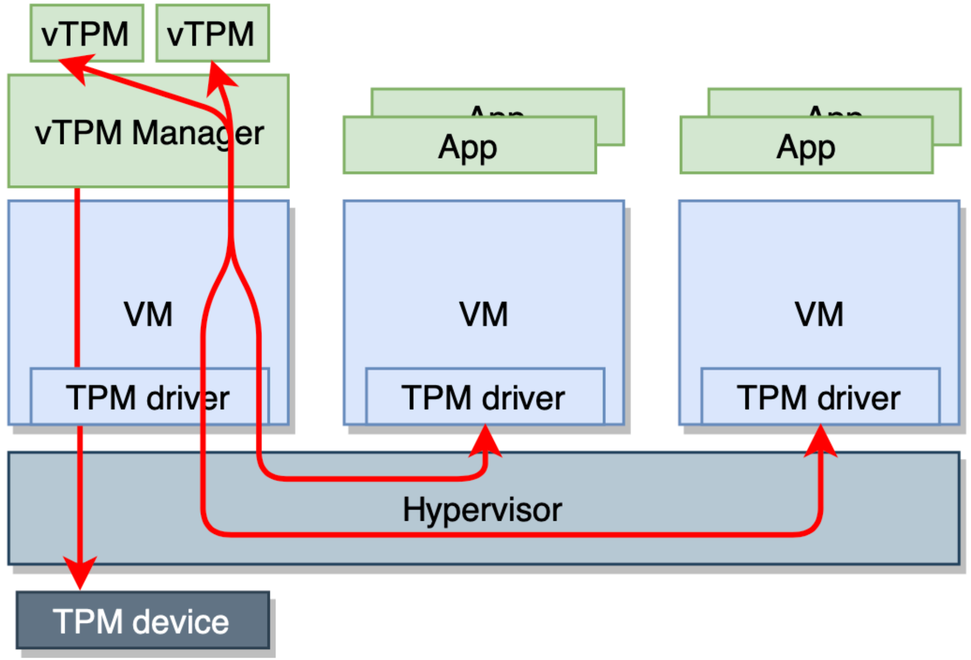
Discussion

Virtualizing the TPM originally proposed in 2006 as example

Cited and adopted widely in related research

Each VM gets vTPM instance

Manager inside a Virtualization System domain “handles” all aspects of vTPM instances



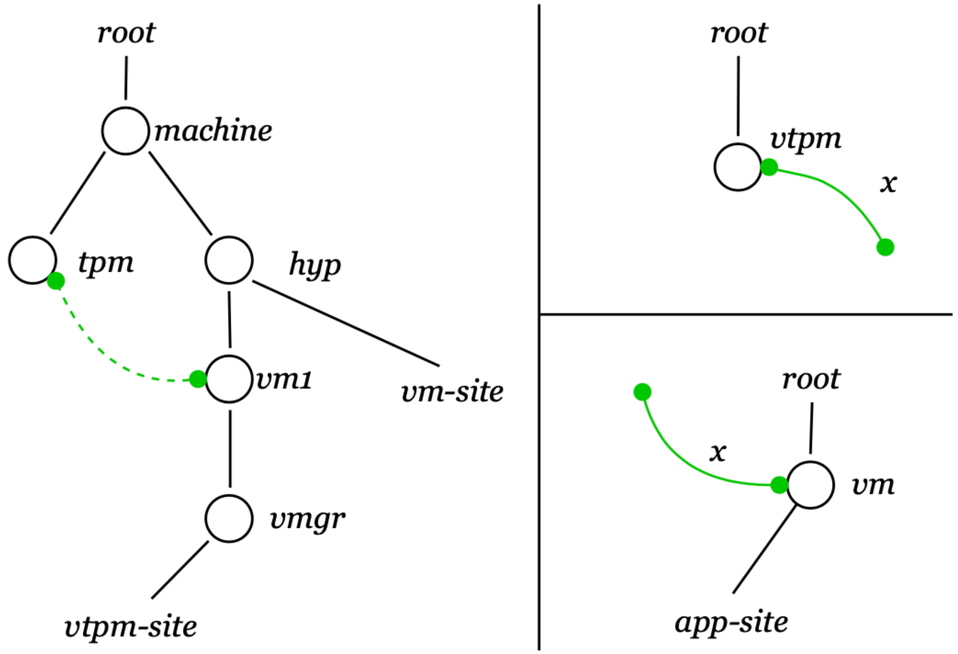
Discussion

Deconstructed into components and hosting systems

We use x to denote open links

A *site* is where components can be placed ...

... open links work as additional interfaces



Discussion

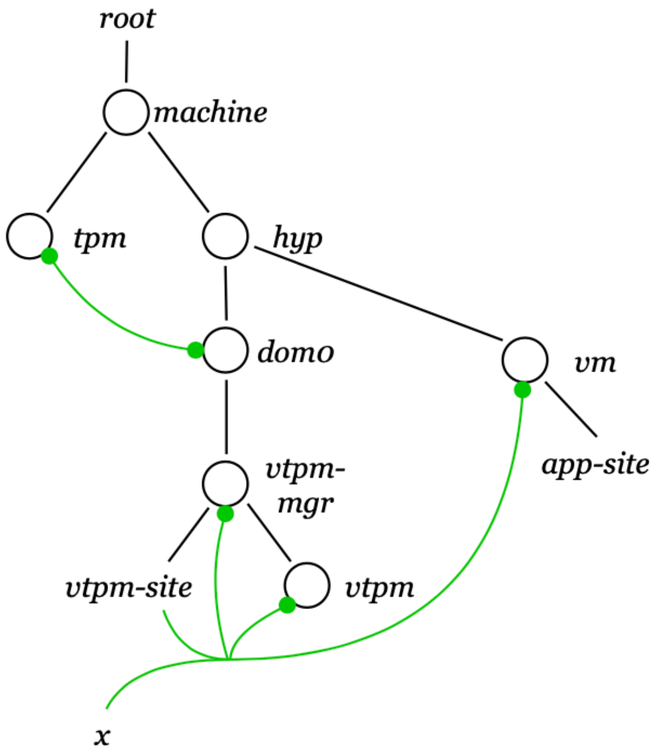
Expressing the system state by assembling the component graph

Adding the vTPM manager into the picture

This shows pictorially (and algebraically) that all possible instances of VMs and vTPMs are joined similarly

—> vTPM-mgr has to be trusted

—> we can not verify the association based on some evidence or information



Discussion

Trust by default

+ No changes, just awareness

- complexity and attack surfaces make this difficult

Remove malware on systems

+ Only need to worry about VM

- Hard

- Circular

- Still need to verify or trust

Remove interfaces

+ May help to prevent dynamic re-provisioning of vTPM

- Still need support to change VM and vTPM location and addresses

Confining Adversaries via Measurements

+ Record co-location of VM and vTPM,

+ Record transactions such as provisioning and migration of vTPM

- Recording *kind* of vTPM (binary) not helpful, we need direct information about instances

- Frequent PCR updates

- reporting of client info to arbitrary challengers

Discussion

Unikernel vTPMs

+ vTPM as standalone component

+ Association deferred to hypervisor

- Additions to hypervisor or involve yet another privileged domain to handle additions

- Cloud functions pushed to hypervisor

vTPM as part of VM abstraction

+ Reduces TCB of vTPM to hypervisor

+ Intuitively fewer possibilities for dynamic modifications

- Aggregation

- vTPMs are controlled by VM and run at hypervisor level

Secure Execution Environment

+ vTPM TCB (sort of) independent from VS

+ Faster attestation infrastructure

- VS still manages connections

- VS info has to be propagated into vTPM

Hardware assisted vTPM

+ Set up VM and hardware manages virtual RoT

+ Hardware assisted vTPM's

- Significant changes

- TPM 2.0 spec supports some use-cases but no full support is currently possible

Conclusion

Several issues with using virtual TPM as root of trust, association is one of them

vTPM can currently be used against a verifier

Architectural support for virtual TPM might be a trustable solution

Summary of Contributions

Highlight the importance of VM-vTPM association with Goldeneye which uses a vTPM against a verifier

Formal framework and model of virtualization systems with hardware roots of trust

We provide an intuitive link between the system model and classical reasoning

We show how related work may fall victim to a Goldeneye attack and discuss solutions

Impact

Scope for TCG Virtualized Platform WG

New version of Virtualized Trusted Platform Architecture Specification

ISO 27070 Security requirements for virtual Roots of Trust

Thank You

hagen.lauer@monash.edu

carsten.rudolph@monash.edu

surya.nepal@data61.csiro.au

amin.sakzad@monash.edu

MONASH University

https://dl.acm.org/citation.cfm?id=3357347